

2 AMIGA

Mark&Technik

DM 16,-
ÖS 120,- /Sfr 16,-
Lit 14000 /hfl 21,- /dkr 75,-

SONDERHEFT

Animation mit Nachbrenner

■ **Zum Abtippen:**
*Sensationelles Tool
für bewegte Grafiken*

Super-Listings

■ **Fast-Load-Copy:**
*Schnelles Directory
durch optimierte Kopien*

■ **Haushaltsbuch:**
*Ihre Finanzen
endlich im Griff*

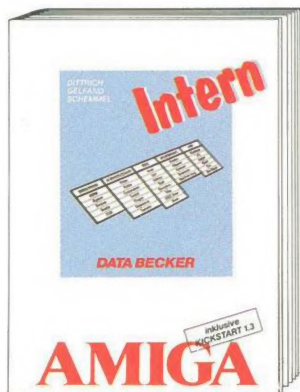
■ **Gravitation:**
*Jonglieren mit Sonnen
und Planeten*

■ **Vokabeltrainer:**
*Intelligentes
Lernprogramm für
mehrere Sprachen*

Tips & Tricks zu Superbase

Alle Programme auch auf Diskette erhältlich

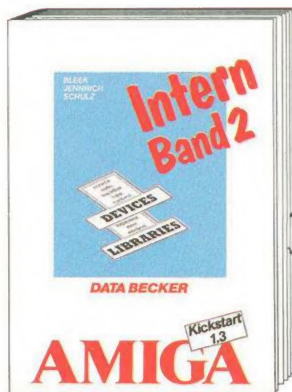
Heute lesen!
was morgen
passiert



Neuaufgabe: die Inside-Story.

Amiga Intern – ein Intern, wie man es von DATA BECKER gewohnt ist. Mit allem, was dazugehört: 68000-Prozessor, CIA, Blitter, Custom-Chips, die Strukturen von EXEC, I/O-Handhabung, Verwaltung der Resources, Erstellung eigener Devices, Exec-Base, resistenteste Programme, Autoboot mit der ROMboot library, DOS-Funktionen, interne DOS-Bibliothek, Aufbau einer Diskette, Programmierung eigener DOS-Handler.

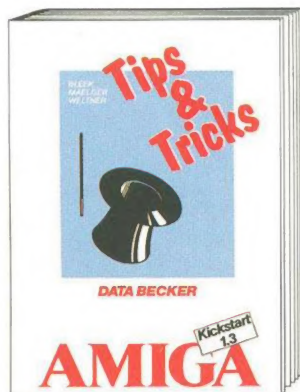
Amiga Intern
Hardcover, 716 Seiten
DM 69,-



Knallharte Informationen.

Amiga Intern Band 2 – das Buch für jeden aktiven Programmierer, der alle weiterführenden Informationen zu seiner Arbeit schnell und zuverlässig finden will: Ein- und Ausgabe über Devices, Standard-Austausch-Formate und Komprimierungsverfahren, alle Amiga-Libraries mit den dazugehörigen Strukturen, Basis- und Grundstrukturen, Preferences als Datenstruktur, Datenübermittlung von Workbench und CLI, Konventionen im Programmierstil und alles zur Version 1.3.

Amiga Intern Band 2
Hardcover, ca. 750 Seiten
DM 69,-
erscheint ca. 11/88



Mit Programmen zaubern.

Mit Amiga Tips & Tricks läßt sich die Arbeit mit Ihrem Rechner noch effektiver gestalten – selbstverständlich bereits unter Berücksichtigung des neuen Betriebssystems (Version 1.3): Gestaltung eigener Programme, Tips & Tricks zum AmigaBASIC, Einbinden von Maschinenprogrammen in AmigaBASIC, Einsatz von DOS-Routinen, optimierende Hilfsprogramme für AmigaBASIC-Programme, Tips zur Arbeit mit der Workbench, Aufbau der Icons, die neuen Preferences.

Amiga Tips & Tricks
Hardcover, 555 Seiten, DM 49,-



Runter von der Workbench.

Rein ins AmigaDOS: Umlenken der Ein- und Ausgabe, mit RAM-Disk und CLI arbeiten, STARTUP-Sequenz, Multitasking mit dem CLI, der interne Aufbau der CLI-Befehle, eigene CLI-Befehle programmieren... Das große Buch zu AmigaDOS – mit nützlichen Batch-Dateien und einer Beschreibung der neuen CLI-Befehle und Devices unter V1.3!

Das große Buch zu AmigaDOS
Hardcover, 370 Seiten
inkl. Diskette, DM 59,-



Alles zur Amiga-Floppy.

Brandaktuell: die zweite, erweiterte Auflage mit allem, was Bezug zur Floppy hat: Workbench, CLI, AmigaBASIC mit verschiedenen Dateitypen, Zugriff aufs Betriebssystem (mit File-Verwaltung, Trackdisk-Device, Boot-Block sowie Checksummen) und direkter Zugriff ohne DOS (MSM- und GCA-Codierung, Track lesen und schreiben, SYNC-Markierung). Dazu einen Floppyspeeder, einen Disketten-Monitor und ein schnelles, leistungsstarkes Kopierprogramm.

Das große Amiga Floppybuch
Hardcover, 560 Seiten
inkl. Diskette, DM 59,-

COUPON!

HIERMIT BESTELLE ICH FÜR MEINEN AMIGA

NAME, VORNAME

STRASSE

ORT

zzgl. DM 5,- Versandkosten unabhängig von der bestellten Stückzahl
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

Ihre Meinung bitte

Alle, die an der Entstehung des Amiga-Sonderheftes mitwirken, haben stets ein Ziel vor Augen: Wir wollen Ihnen den Umgang mit Ihrem Computer erleichtern. Sie sollen in jedem Heft viele interessante Themen finden. Die vorgestellten Programme schließen Lücken im Software-Angebot, sind eine interessante und erschwingliche Alternative zu kommerziellen Produkten. Die Schwerpunkte dieses Sonderhefts sind nach langen Diskussionen ausgewählt worden. Die Texte, die Sie auf den folgenden Seiten lesen werden, sind überprüft, geändert, nochmals überprüft und nochmals geändert worden. Die Entstehung des Heftes, das Sie jetzt in den Händen halten, hat ungefähr drei Monate gedauert. Sind Sie mit dem zweiten Amiga-Sonderheft zufrieden, haben wir die richtige Themenwahl getroffen? Sagen Sie uns bitte Ihre Meinung. Gelegenheit dazu geben wir Ihnen mit der Leserumfrage auf Seite 14. Schreiben Sie uns, welche Themen Sie besonders interessie-

ren. Wünschen Sie sich in den folgenden Sonderheften Kurse und Listings? Sollen wir auf lange Listings im Heft völlig verzichten, sie lieber nur auf der Programmservice-Diskette veröffentlichen?

Ihre Meinung ist für uns wichtig, um in kommenden Ausgaben noch interessantere Themen für Sie auszuwählen. Übrigens: Die Sonderhefte werden ab 1989 regelmäßig im Abstand von zwei Monaten erscheinen.

Aber zusätzlich zum Fragebogen steht Ihnen natürlich jederzeit frei, per Brief, Postkarte oder telefonisch Stellung zu den Sonderheften zu nehmen und Fragen an die Redaktion zu stellen.

Wenn Sie anrufen möchten, beachten Sie bitte unsere regelmäßige Sprechzeit: An jedem Donnerstag sind wir von 14 bis 17 Uhr zu erreichen. Gemeinsam mit Ihnen haben wir die Chance, sehr hohen Ansprüchen zu genügen. Nutzen Sie diese Möglichkeit, wir geben unser Bestes.

Ihr
Ralf Sablowski
Redakteur



Ralf Sablowski

GRAFIK

6 EXTRAKLASSE: SPRITE-EDITOR

Bewegte Objekte sind das Salz in der Suppe — das gilt vor allem für Computer-Spiele. Entwerfen Sie mit dem »Object-Editor« bunte Figuren, die dann im Handumdrehen animiert werden.

SPIELE

10 HEISSE SCHLACHTEN IM AMIGA

Lieben Sie Risiko? Hier ist nicht die Neigung zu waghalsigen Taten gemeint, sondern das gleichnamige Brettspiel. Diese Umsetzung ist einfach super, erleben Sie das bekannte Strategie- und Glücksspiel auf dem Amiga.

119 SCHACH DER LANGEWEILE

Shogun ist der Titel japanischer Feldherren, die anstelle der machtlosen Kaiser das Land regieren. In einem dem Schach ähnlichen Spiel versetzen wir Sie in die Zeit der Shogun zurück.

LESERUMFRAGE

14 SAGEN SIE UNS IHRE MEINUNG

Was erwarten Sie von »Ihrem« Amiga-Sonderheft? Liegt Ihr Hauptinteresse bei Kursen, Listings oder Tests? Gestalten Sie Ihr Sonderheft mit, füllen Sie den Fragebogen aus. Zu gewinnen gibt es dabei natürlich auch etwas.

ANWENDUNGEN

18 FINANZEN IM GRIFF

Unser elektronischer Finanzmanager hilft Ihnen, Einnahmen und Ausgaben ständig unter Kontrolle zu behalten. Überprüfen Sie Ihren Etat mit diesem komfortablen Programm.

23 INTELLIGENTER SPRACHENTRAINER

Vor dem Beherrschen einer Sprache sind Vokabeln zu lernen. Vermeiden Sie in Zukunft langweilige Büffelei, verbinden Sie Spaß und Lernen mit dem Vokabeltrainer.

29 MEISTER DER TASTEN

Vom Zehn-Finger-System haben Sie sicher schon gehört. Möchten Sie ein Virtuose auf der Computer- oder Schreibmaschinentastatur werden? Der »Keyboard-Master« führt Sie in zwanzig Lektionen zu diesem Ziel.

GRAFIK-TOOLS

36 BILDER LERNEN LAUFEN

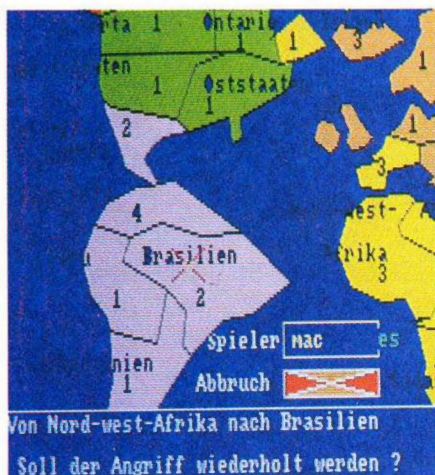
Filmähnliche Sequenzen lassen sich bisher nur mit teuren Programmen erstellen. Wir bieten eine Alternative: Zeichnen Sie Bilder mit einem Malprogramm, fügen Sie diese dann mit Hilfe von »PlayAnim« zu einem Trickfilm zusammen.

51 FONTS MIT ALLEN TRICKS

Verschiedene Schriftarten, sogenannte Fonts, meistert der Amiga lässig. Nur wenige Programme unterstützten jedoch bisher das Konstruieren eigener Zeichensätze. Der Fontdesigner schließt diese Lücke — und das »fontastisch«.

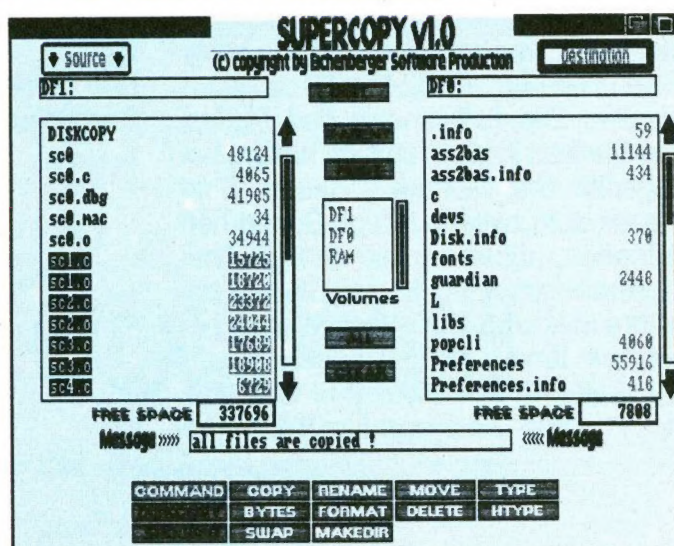
Behalten Sie Einnahmen und Ausgaben mit dem Amiga unter Kontrolle. Mit dem elektronischen Haushaltsbuch bekommen Sie Ihre Knete in den Griff — und das mit viel Komfort.

SEITE 18



In diesem Spiel geht es um Karten, Kontinente und Kanonen. Erobern Sie die Spielwelt mit Taktik und Glück.

SEITE 10



Der Umgang mit dem CLI hält nicht, was die grafische Benutzeroberfläche des Amiga verspricht. Führen Sie CLI-Operationen von nun an bequem mit »Supercopy« durch.

SEITE 90

90 AMIGA KUNTERBUNT

Einzelne Bildteile kopieren und verfremden, das ist mit dem Amiga kein Problem. Wollen Sie bei eigenen Programmen die Manipulationen auf eine einzelne Farbe beschränken, bieten die Bibliotheken des Betriebssystems keine Unterstützung. Mit dieser Routine wird das anders.

SIMULATION

74 DER WELTENBAUER

Diese Simulation verdeutlicht spielerisch die komplexen Zusammenhänge der Gravitation. Lernen Sie das Zusammenspiel gewaltiger Kräfte kennen, gestalten Sie Planetensysteme jetzt selbst!

FLOPPY-TOOLS

81 »FLOPPY-SPEEDER« MIT NIVEAU

Nervt Sie das langsame Laden des Inhaltsverzeichnisses der eingelegten Diskette? Räumen Sie doch einfach Ihre Disketten auf. »FastLoadCopy« schafft Ordnung auf Floppy-Disks und dient gleichzeitig als Kopierprogramm.

90 KOPIEREN MIT VIELEN EXTRAS

»CLI-Mate«, »Zing!« und »DiskMaster« sind Ihnen sicher ein Begriff. Wir präsentieren ein leistungsstarkes Tool, das sich mit diesen Programmen messen kann.

TIPS & TRICKS

112 EIN COCKTAIL FÜR ALLE FÄLLE

Dieser Cocktail wird Ihnen schmecken: Zahlreiche Tips, die das Leben eines Amiga-Anwenders erleichtern. Verblüffen Sie Freunde mit den neu erworbenen Kenntnissen, beispielsweise mit dem Geheimnis der blinkenden Power-LED.

117 TIPS ZU SUPERBASE PROFESSIONAL

Superbase Professional und SB 2 gehören zu den leistungsfähigsten Dateiverwaltungen. Lernen Sie neue Seiten dieser Programme kennen.

BÜCHER

124 FUTTER FÜR LESERATTEN

Übersicht über ergänzende Literatur zu den Schwerpunkten dieses Sonderhefts.

EINGABEHILFEN

159 CHECKSUMMER

Wie gebe ich Programme ein? Diesen Artikel sollten Sie unbedingt lesen, wenn Sie Programme aus diesem Sonderheft abtippen möchten.

SONSTIGES

3 EDITORIAL

162 IMPRESSUM

162 INSERENTENVERZEICHNIS


Alle Programme aus Artikeln mit einem -Symbol finden Sie auch auf der Programmservice-Diskette zu diesem Sonderheft

Mit dem Objekt-Editor erstellen Sie Sprites für eigene Spiele — Animation inklusive.

SEITE 6

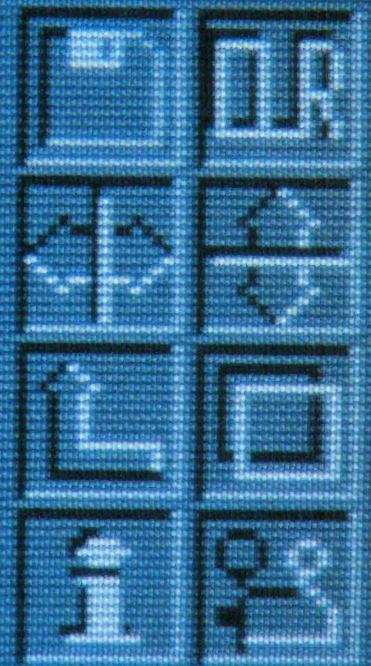
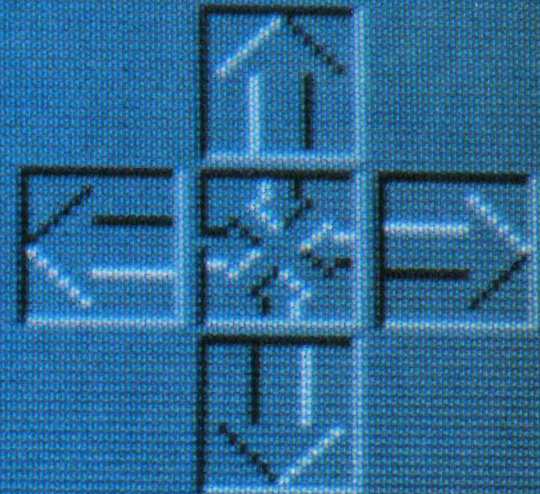
Beteiligen Sie sich an der Leser-Umfrage und gewinnen Sie ein Doppel-Laufwerk mit Overdrive-System-Port.

SEITE 14



Sprite-Editor

Sprites gehören zu Spielen wie der Monitor zum Computer. Leider müssen diese auch auf dem Amiga in mühevoller Kleinarbeit erstellt werden. Unser Sprite-Editor beseitigt dieses Problem und macht selbst das Entwickeln von überdimensionalen Sprites zum Kinderspiel.



Objekt Nr.

001



Es gibt kaum Spiele, die ohne Sprites auskommen. Dabei beschränkt sich die Verwendung von bewegten Objekten keineswegs nur auf Actionspiele. Auch Brettspiele, Knobelspiele, Adventures etc. profitieren ungemein durch den Einsatz beweglicher Objekte. Mit der mausgesteuerten Benutzeroberfläche des »Amiga-Object-Editor« (Bild 1) lassen sich Sprites und Animationssequenzen schnell und mühelos erstellen und editieren. Auch DPaint-Brushes lassen sich damit bearbeiten.

Geben Sie den DATA-Lader »AOE_Gen« (Listing 1) mit dem Checksummer ein. Da das Programm einen relativ hohen Speicherbedarf hat, muß vor der Eingabe des Listings im Direktmodus des Basic-Editors »CLEAR, 150000« eingegeben werden. Dies hat auch dann zu geschehen, wenn »AOE_Gen« erneut in den Speicher geladen wird. Nach dem Start des Laders generiert dieser das lauffähige Programm »AOE_V1.2« auf Diskette. Nach diesem Vorgang, der etwas Zeit beansprucht, kann der Object-Editor gestartet werden. Dabei ist zu beachten,

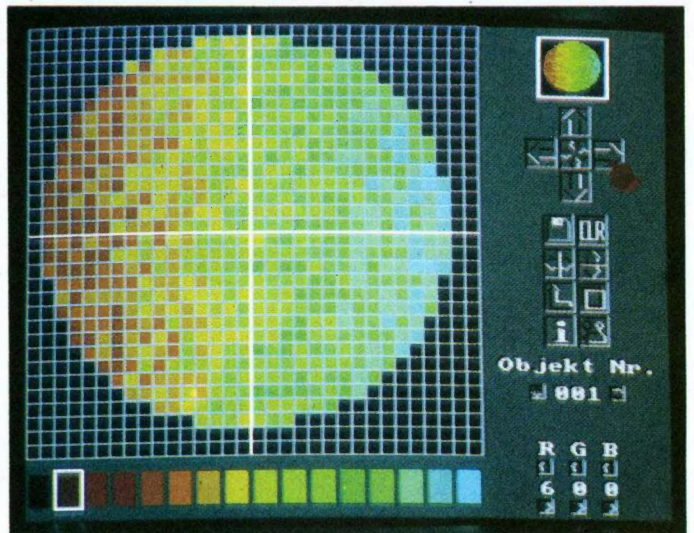


Bild 1. Die Benutzeroberfläche des »Amiga-Object-Editors« mit den verschiedenen Icons

daß das vom Lader generierte File nur vom CLI aus gestartet werden kann, obwohl ein Icon vorhanden wäre.

Sie werden nach dem Start aufgefordert, die Anzahl der Objekte, die bearbeitet werden sollen, einzugeben. Klicken Sie dazu das Eingabefeld an und tragen die gewünschte Objektanzahl ein.

Das Zeichnen eines Objektes ist durch die Maussteuerung sehr einfach. Es stehen 16 verschiedene Farben zur Verfügung, die durch einfaches Anklicken ausgewählt werden. Die Rot-Grün- und Blauanteile der gewählten Farbe können durch die Schalter am unteren rechten Bildschirmrand verändert werden.

stellt. Rechts oben sieht man das Sprite in Originalgröße. Unter diesem Fenster befinden sich vier Pfeile. Durch Anklicken dieser Pfeile kann das aktuelle Sprite in alle vier Himmelsrichtungen verschoben werden. Klicken Sie das Icon in der Mitte der Pfeile an, so ändern diese ihre Richtung und das Sprite wird nun bei der Verwendung der Pfeilsymbole in Richtung seiner Mittelachse gestaucht. Unter den Pfeil-Icons befinden sich acht weitere Icons, auf deren Funktionen nun eingegangen werden soll: **Icon 1 (Disketten-Icon):** Hier haben Sie die Möglichkeit, Sprites zu laden und zu speichern. Zum Laden muß die entsprechende Option gewählt

Adresse	Wert	Bemerkung
\$00000000	"AOE ".L	Kennung der Objektdatei
\$00000004	0000.W	Anzahl der Objekte in der Datei
\$00000006	0000.W	Nun folgen 16 Wörter mit der Farbinformation
\$00000026		Es folgen die Objektdaten: Anzahl der Objekte mal 640 Byte.
Die Objektdaten haben folgenden Aufbau:		
1. Objekt:		1. Bitplane (128 Byte) 4. Bitplane (128 Byte) 5. Maske (128 Byte)
2. Objekt:		1. Bitplane (128 Byte)
Jedem gesetzten Bit in der Maske entspricht ein gesetzter Punkt im Sprite		

Tabelle 1. Der Aufbau eines Files im »AOE«-Format

Gezeichnet wird ein Sprite mit Hilfe der Maus, wobei mit der linken Maustaste Punkte gesetzt und mit der rechten gelöscht werden. Das zu bearbeitende Sprite wird bei »Objekt Nr.« ausgewählt.

Im Zeichenfenster wird das Riesensprite, das in Wirklichkeit aus vier Einzelsprites besteht, stark vergrößert darge-

und die Nummer, unter der das Objekt im Speicher stehen soll, sowie der Filename eingegeben werden.

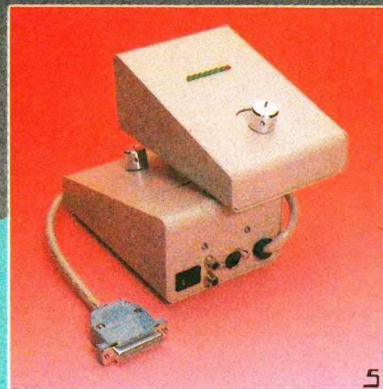
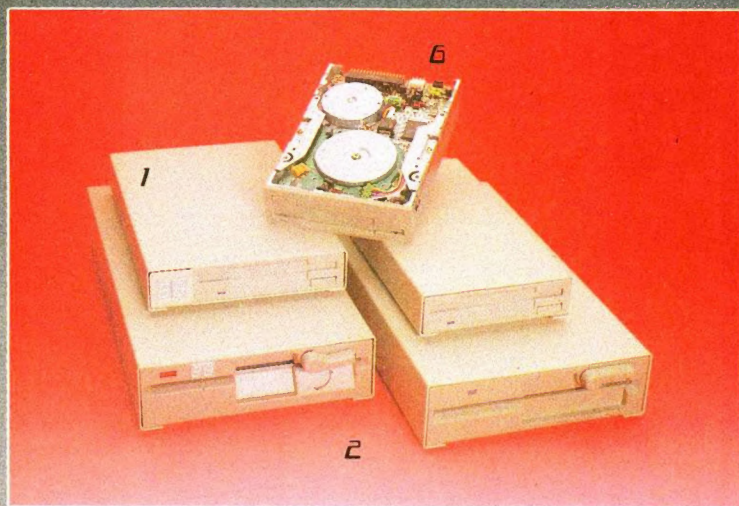
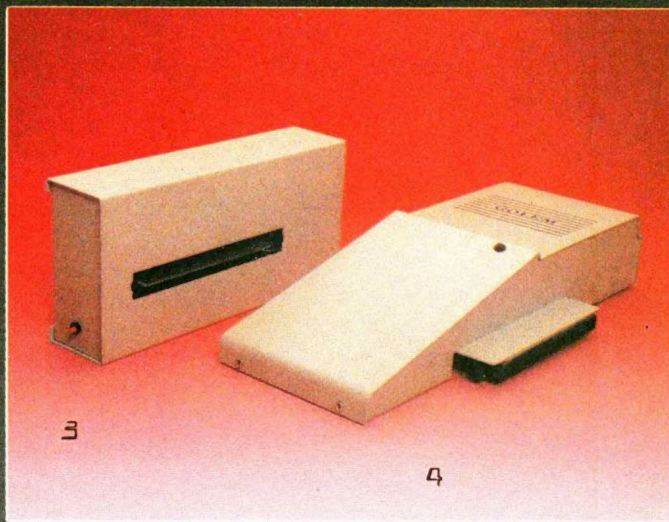
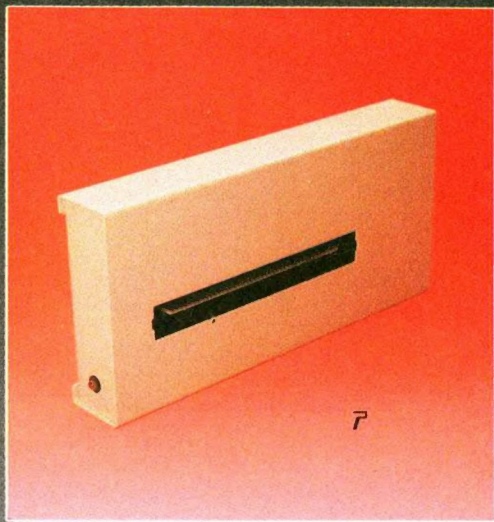
**Fortsetzung
des Artikels
auf Seite 126**

KUPKE



02 31/81 83 25-27
Telefax 02 31/81 74 29
D-4600 Dortmund 1
Burgweg 52 a

GOLEM



**Wir
liefern im
3-Tage-Rhythmus**

1 Golem Drive 3,5 Display

NEC 1037a mit heller Frontblende ● Amiga-farbenes Metallgehäuse ● Abschalte ● Busdurchführung bis DF3 ● Sidecar, PC 1 und PC Karten-kompatibel ● Trackdisplay zur aktuellen Spur- und Kopfanzeige

mit Display
ohne Display

DM 359.-
DM 339.-

2 Golem Drive 5,25 Display

NEC Laufwerk mit heller Frontblende ● Amiga-farbenes Metallgehäuse ● Abschalte ● 40/80 Track-Umschalte ● Busdurchführung bis DF3 ● PC Karten, Sidecar und PC 1 kompatibel ● Trackdisplay zur aktuellen Spur- und Kopfanzeige

mit Display
ohne Display

DM 449.-
DM 419.-

3 Golem Ram Box 2MB

2MB Speichererweiterung für den Amiga 1000 ● ansteckbar am Systembus ● Abschalte ● Busdurchführung ● autokonfigurierend ● Betriebskontrollanzeige ● Amiga-farbenes Metallgehäuse ● erweitert den Grundspeicher auf 2,5 Megabyte

komplett
ohne Ram's

DM 1398.-
DM 449.-

4 Golem 500 Ram Box

2MB Speichererweiterung im formschönen 500'er Design ● Busdurchführung ● autokonfigurierend ● Betriebskontrollanzeige ● externer Anschluß an den Systembus ● erweitert den Grundspeicher auf 2,5 Megabyte

komplett
ohne Ram's

DM 1398.-
DM 449.-

Technische Änderungen vorbehalten

5 Golem Sound Stereo

Audio Digitizer der Spitzenklasse ● kompatibel zur meisten Samplersoftware ● DIN- und Cinch Anschluß auch für Micro geeignet ● optisches Aussteuerungsdisplay ● Stereowandlung ● umschaltbar auf Mono-Betrieb

Stereo

Mono ohne Display

DM 189.-
DM 139.-

6 Golem Drive A 2000

internes Amiga Drive ● NEC 1036a mit heller Frontblende ● einbaufertig modifiziert ● mit Staubschutzklappe ● incl. Einbauanleitung und Montagesatz

DM 200.-

7 Kickstart/Uhrenmodul

"Bitte Workbench einlegen", meldet ihr Amiga 1000 nach dem Einschalten mit dem extern ansteckbaren Kickstartmodul ● Busdurchführung ● Abschalte, so daß andere Kickstartversionen wieder gebootet werden können ● alle gängigen Kickstart-Versionen lieferbar

DM 199.-

Amiga 500/2000 kompatibles Uhrenmodul ● Akkugepuffert ● extern ansteckbar

im Extragehäuse

Uhr u. Kick in einem Gehäuse

DM 149.-
DM 299.-

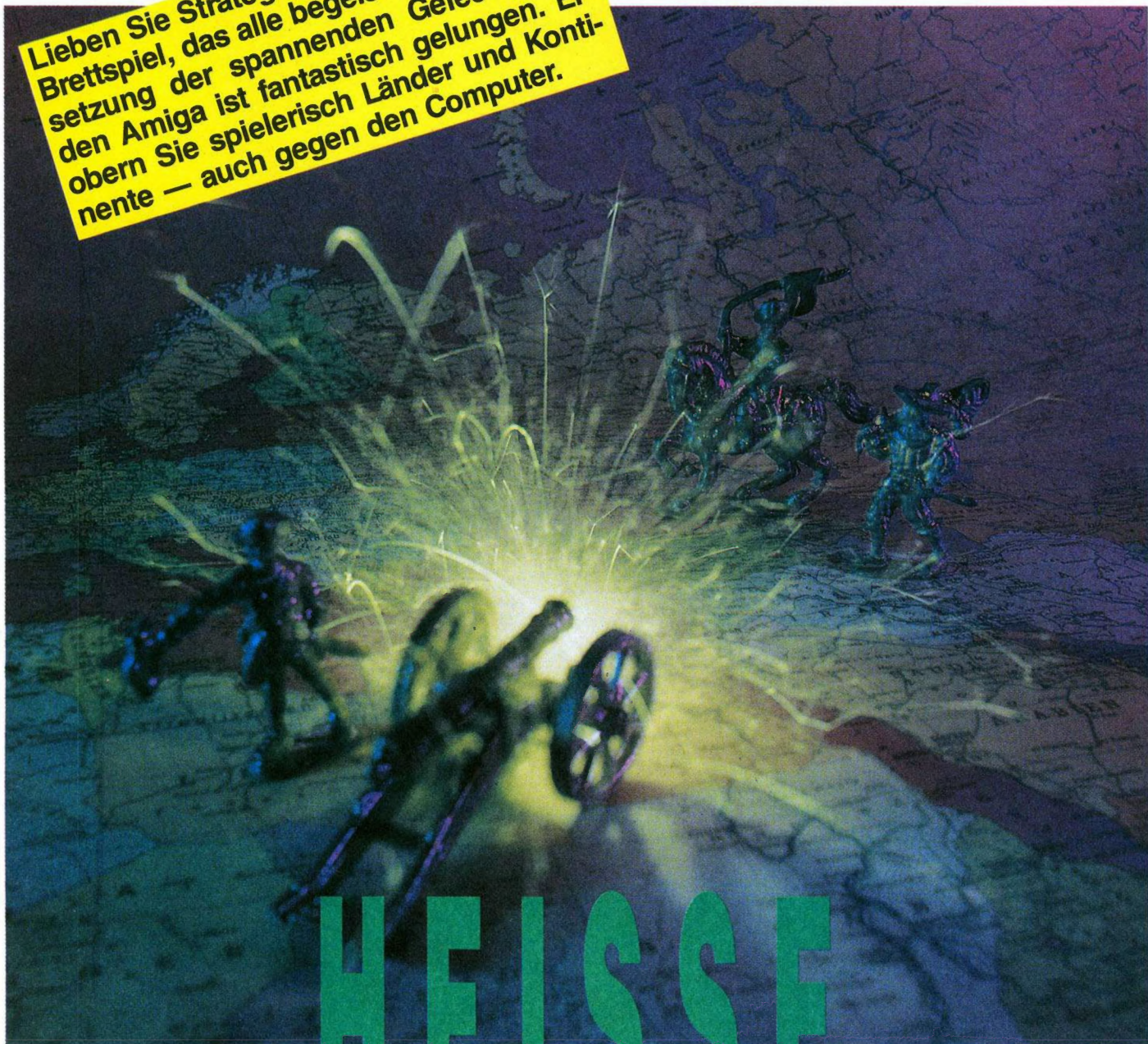
8 Kickstartumschaltplatine

Intern einsteckbare Umschaltplatine bestückt mit einem zusätzlichen Kickstart ● alle gängigen Versionen ● keine Lötarbeiten erforderlich ● umschaltbar auf original Kickstart

komplett
ohne Eprom's

DM 149.-
DM 59.-

Lieben Sie Strategiespiele? Risiko ist ein Brettspiel, das alle begeistert. Diese Umsetzung der spannenden Gefechte auf den Amiga ist fantastisch gelungen. Erleben Sie spielerisch Länder und Kontinente — auch gegen den Computer.



HEISSE Schlachten im Amiga

Die Blauen sind auf dem Vormarsch. Australien wurde bereits im Handstreich erobert, Asien ist zu zwei Dritteln von ihnen besetzt. Die Gelben haben sich in Nord-Amerika eingenistet, auch sie sind eine große Bedrohung. Der Gegner konzentriert seine Armeen im Mittleren Osten. Jetzt greift der Feldherr der blauen Steine mit einer gewaltigen Armee Europa an, Ihre einzige Bastion wackelt. Werden Sie diesen gewaltigen Zweikampf für sich

entscheiden? So könnte die Lage im Verlaufe des Strategiespiels »Amiga gegen den Rest der Welt« aussehen. Diese Umsetzung eines beliebten Brettspiels der Firma Parker ist faszinierend. Jeder, der einmal

Amiga gegen die Welt

mit ein paar Freunden um die Welt gestritten hat, wird das bestätigen. Natürlich kann auch Ihr Amiga direkt in den Kampf eingreifen. Bei maximal

sechs Mitspielern in einem Spiel sind bis zu fünf Computer-Gegner zu aktivieren.

Wenn Sie in den Genuß des Strategiespiels kommen wollen, geben Sie bitte Listing 1 (»Lader«) und das Hauptprogramm (Listing 2) mit dem Checksummer ein. Hinweise zu dieser Eingabehilfe finden Sie auf Seite 159 in diesem Heft.

Haben Sie die Programme auf Diskette gespeichert, kann der Spaß beginnen. Laden Sie

Amiga-Basic und starten von dort aus das Programm »Lader«. Der Speicher für das Basic-Programm wird jetzt auf 40000 Byte vergrößert und das Hauptprogramm geladen.

Die erste Meldung des Spiels fragt nach der Teilnehmerzahl der menschlichen Spieler. Wählen Sie eine Zahl von 0 bis 6, schließen Sie diese und alle folgenden Eingaben bitte mit <RETURN> ab. Jetzt möchte das Programm noch wissen, wie viele Gegner der Computer übernehmen

soll. Die erlaubten Eingaben sehen Sie in der Klammer hinter der Frage.

Anschließend sind für die Spieler (menschlich) Namen einzugeben, maximal acht Buchstaben sind jeweils erlaubt. Die Wahl der Namen für die computer-gesteuerten Mitspieler übernimmt das Basic-Programm. Sie lauten beispielsweise »Amiga« und »Atari«, der immerwährende Konkurrenzkampf zwischen diesen beiden Firmen wird scheinbar jetzt mit anderen Waffen fortgesetzt.

Schließlich wählt jeder Spieler aus einer vorgegebenen Tabelle eine ihm genehme Farbe (Farbnummer eingeben!) aus, und los geht's.

Verteilung der Länder

Was ein rechter Feldherr ist, der herrscht natürlich über eine Reihe von Ländern. Insgesamt ist die Spielwelt in 42 fiktive Staaten aufgeteilt. Um nicht gleich zu Beginn des Spiels einen Streit vom Zaun zu brechen, werden jedem Mitspieler rein zufällig Länder vom Programm zugeteilt. Die 42 Staaten werden also durch die Zahl der Mitspieler geteilt und die jedem zustehende Anzahl vergeben. Sie erkennen Länder eines Feldherren an der gleichen Farbe (Bild 1).

Jetzt startet das Spiel, in dem es für alle um das gleiche Spielziel geht: Erobern Sie die Welt.

Ein zufällig gewählter Spieler beginnt die Aktionen. Doch wie soll man mit Staaten andere Länder erobern? Natürlich benötigt man dazu Streitkräfte, die in unserem Spiel als »Armeen« bezeichnet werden. Jeder Spieler erhält für jedes Land anfangs eine Armee. Ist ein Spieler nun an der Reihe, erhält er nach einer einfachen Formel neue Armeen hinzu. Die Formel lautet:

Besetzte Länder / 3
= neue Armeen

Das Minimum an zugeteilten neuen Armeen ist jedoch immer drei, auch wenn ein Spieler vielleicht nur noch ein Land besitzt.

Wir stehen nun am Anfang des Kampfes um die Spielwelt. Sechs Teilnehmer warten auf die ersten Schwächen ihrer Gegner. Der erste Spieler erhält drei äußerst muntere Armeen und steht vor der Frage, in welches seiner Länder er diese nun einquartiert. Die Far-

be des Spielers erkennen Sie an der Figur unten rechts. Ist die Entscheidung getroffen, bewegt er den Mauszeiger auf dieses Land (es muß natürlich schon ihm gehören) und klickt es mit der linken Maustaste an. Der Spieler entscheidet weiterhin, wie viele seiner Armeen er in dieses Land positioniert. Eine beliebige Aufteilung der Streitkräfte ist regelgerecht, jedoch anfangs nicht sinnvoll.

Sind die verteilten Armeen einquartiert, ändert sich die Zahl in dem betreffenden Land. Sie zeigt nun die aktuelle Anzahl der Armeen.

Gut, jetzt hat unser Spieler seine Armeen in ein Land verfrachtet. Und jetzt? Alle Spieler haben ja die Aufgabe, die Welt mit den 42 Staaten zu erobern. Fangen wir also am besten damit an, ein neues Land zu erkämpfen.

Aktivieren Sie die Menüleiste mit der rechten Maustaste. Sie finden oben links auf dem Bildschirm das Menü »Aktionen«. Fahren Sie bei gedrückter rechter Maustaste auf die-

Land 1	Land 2
Alaska	Kamtschatka
Grönland	Island
Island	Skandinavien
Island	Großbritannien
Großbrit.	Skandinavien
Großbrit.	Westeuropa
Mittl. Osten	Ostafrika
Mittl. Osten	Ägypten
Südeuropa	Mittl. Osten
Brasilien	N/W-Afrika
Südeuropa	N/W-Afrika
Südafrika	Madagaskar
Indonesien	Siam
Siam	Neu-Guinea
Neu-Guinea	Indonesien
Neu-Guinea	Ost-Australien
Indonesien	West-Australien
Indonesien	Ost-Australien

Tabelle 1. Diese Länder sind auch Nachbarländer

Kontinent	Armeen
Asien	7
Nordamerika	5
Europa	5
Afrika	3
Südamerika	2
Australien	2

Tabelle 2. Diese Anzahl an Armeen gibt es zusätzlich für besetzt gehaltene Kontinente

gewählt, in dem mehr als eine Armee steht, fragt das Programm nach dem Ziel Ihrer Wünsche, also nach dem anzugreifenden Land. Da wir es ja mit soliden Armeen zu tun haben, die nicht fliegen können, ist ein Land auszuwählen, das direkt benachbart ist. Dieses ist ebenfalls anzuklicken, und schon tobt eine Schlacht um das Zielland.

In Tabelle 1 finden Sie die Übergänge zwischen Ländern, die nicht direkt aus der Karte deutlich werden.

Die Entscheidung, welcher Feldherr zukünftig das angegriffene Land zu seinem Besitz zählen darf, fällt jetzt. Angenommen, Sie sind der Angreifer und haben vier Armeen im Basisland. Der Verteidiger besitzt nur eine einzige Armee in dem angegriffenen Land. Die Entscheidung über Sieg oder Niederlage in einer Schlacht fällt mittels simulierten Würfelergebnissen. Der Angreifer besitzt vier Armeen, drei davon sind für den Angriff einzusetzen. Die drei Angriffsarmeen



Bild 1. Erobern Sie die Welt, eine farbenfrohe Karte dient als Spielfeld

ses Menüfeld. Ein Fenster klappt auf, in dem Sie fünf Unterpunkte sehen:

Angriff

Wählen Sie bei gedrückter rechter Maustaste den Punkt Angriff und lassen die Taste los.

Im unteren Bereich des Spielfelds erscheint das Wörtchen »Von«. Bewegen Sie nun den Mauszeiger auf Ihr Land,

von dem Sie den Angriff starten wollen. Klicken Sie dieses mit der linken Maustaste an. Sollte nur eine Armee in dem gewählten Land stehen, so meldet das Programm »Hier gibt es keine Angriffsarmeen!«. Wir lernen daraus, daß mindestens zwei Armeen in einem Land stehen müssen, von dem aus ein Angriff erfolgen soll. Haben Sie ein eigenes Land

werden in der Entscheidung durch drei Würfel symbolisiert. Die verteidigende Streitmacht des Gegners, eine Armee, stellt einen Würfel. Das Verhältnis bei diesem Angriffswurf lautet also 3:1. Nun wird gewürfelt, das höchste Einzelergebnis eines Wurfes entscheidet. Die Würfelergebnisse sind am rechten unteren Bildschirmrand in einem Kasten darge-



Bild 2. Der Spieler »pc« ist an der Reihe, er besitzt die Länder mit der Farbe, die der Soldat unten rechts zeigt



Bild 3. Alea jacta est — Die Würfel sind gefallen. Wird der Angreifer zum erfolgreichsten Feldherren?



Bild 4. Symbole für die Karten sind Soldat, Reiter und Kanone — eine Serie bringt Zusatzarmeen

stellt. In unserem Fall zeigt dieser Kasten drei Ergebnisse für den Angreifer und einen Wurf für den Verteidiger. Die Ergebnisse werden jetzt miteinander verglichen. Hat der Angreifer als höchsten Wurf eine größere Punktzahl als der Gegner, zieht der Aggressor in das

Land mit den drei Angriffsarmeen ein, die verteidigende Streitkraft wird vom Spielfeld entfernt. Bei Gleichheit der höchsten Punktzahl verliert der Angreifer, er ist also leicht im Nachteil. Hat der höchste Wurf des Angreifers weniger ergeben als der Würfel des

Verteidigers Augen zeigt, verliert der Angreifer selbstverständlich auch. Haben Sie noch mehr als eine Armee im Ausgangsland, werden Sie gefragt, ob Sie den Angriff wiederholen wollen. Bitte klicken Sie dann das Feld »Ja« an.

Im späteren Spielverlauf kann es zu einer Ansammlung von Armeen kommen, die dann in gewaltigen Schlachten um Länder kämpfen. In einem Wurf kann jedoch maximal mit je drei Würfeln »gekämpft« werden. In dem Fall, daß der Angreifer und der Verteidiger mehr als einen Würfel (also eine Armee) einsetzen, gilt folgende Regel:

Die Würfel beider Kontrahenten werden nach der Höhe sortiert, und dann jeweils miteinander verglichen. Bei Gleichheit gewinnt wieder der Verteidiger. Ein Beispiel: Wurf Angreifer: 6, 4, 1 Wurf Verteidiger: 6, 3, 2

Das Ergebnis sagt aus, daß der Angreifer zwei Armeen verliert und der Verteidiger eine. Die Sechsen waren gleich, der Verteidiger gewinnt. Die Vier des Angreifers war höher als die Drei des Verteidigers, eine Armee dieses Spielers muß vom Schlachtfeld. Die Zwei des Verteidigers übertrumpft wieder die Eins des Angreifers, damit verliert dieser seine zweite Armee. Setzt der Verteidiger nur eine Armee ein, verliert der Angreifer bei einem niedrigen Wurf immer höchstens eine Armee, bei zwei Verteidigern höchstens zwei.

Die Berechnungen übernimmt in unserem Spiel natürlich der Amiga.

War der Angriffszug erfolgreich und die letzte Armee des Gegners geschlagen, rücken die siegreichen Armeen in das Zielland ein. Jetzt kann der Spieler, der an der Reihe ist, entscheiden, ob er

- weiter angreift,
- Armeen verschiebt,
- Karten tauscht, oder
- dem nächsten Spieler die Aktionen überläßt.

Die Spielwelt ist in die Kontinente Nordamerika, Südamerika, Europa, Asien, Afrika und Australien unterteilt. Der Besitz eines Kontinentes über eine ganze Runde hinweg wird belohnt. Der Spieler erhält vor seinen Aktionen die normale Anzahl der neuen Armeen. Zusätzlich gibt es als Bonus für den gehaltenen Kontinent eine bestimmte Anzahl. Die Werte entnehmen Sie bitte Tabelle 2. Es ist also ratsam, schnell einen Kontinent zu erobern, um

Deluxe: Software für den Amiga

Deluxe Paint II (deutsch)/Print I
Dieses Grafikprogramm ist eines der außergewöhnlichsten auf dem Softwaremarkt. Jetzt mit Print I.

Bestell-Nr. 54114

DM 199,-* (sFr 179,-*/öS 1990,-*)

Die ideale Ergänzung zu Deluxe Paint II:

Seasons & Holidays

Bestell-Nr. 52580

DM 29,-* (sFr 26,-*/öS 290,-*)

Deluxe Art Parts II

Bestell-Nr. 52581

DM 29,-* (sFr 26,-*/öS 290,-*)

Deluxe Video 1.2 (deutsch)

Mit Deluxe Video können Sie animierte Grafiksequenzen einfach entwerfen und zusammenstellen.

Bestell-Nr. 52583

DM 249,-* (sFr 225,-*/öS 2490,-*)

Deluxe Photolab (deutsch)

Integriertes Grafikpaket und Druckprogramm mit Posterdruckfunktion und einer Vielzahl weiterer erstaunlicher Funktionen.

Bestell-Nr. 54112

DM 249,-* (sFr 225,-*/öS 2490,-*)

Für alle, die nicht auf die deutsche Version warten wollen:

Deluxe Photolab (englisch)

Bestell-Nr. 54117

DM 199,-* (sFr 179,-*/öS 1990,-*)

Deluxe Music (deutsch)

Das professionelle Musikprogramm. Jetzt mit deutscher Software.

Bestell-Nr. 52579

DM 199,-* (sFr 179,-*/öS 1990,-*)

Die ideale Ergänzung zu Deluxe Music:

It's only Rock'n'Roll

Bestell-Nr. 54115

DM 29,-* (sFr 26,-*/öS 290,-*)

Hot & Cool Jazz

Bestell-Nr. 54116

DM 29,-* (sFr 26,-*/öS 290,-*)

Deluxe Productions

(englisch/NTSC)

Bestell-Nr. 54113

DM 399,-* (sFr 359,-*/öS 3990,-*)

Updates von der englischen auf die deutsche Version:

Paint II, Bestell-Nr. 54114U

Video 1.2, Bestell-Nr. 52583U

Photolab, Bestell-Nr. 54112U

je DM 49,-* (sFr 49,-*/öS 490,-*)

Gegen Einsendung der Originaldiskette und gegen Vorauskasse.

In Vorbereitung:

Deluxe Print II (deutsch)

Bestell-Nr. 52582

DM 199,-* (sFr 179,-*/öS 1990,-*)

* Unverbindliche Preisempfehlung

Fragen Sie Ihren Händler nach weiteren Informationen.

Markt & Technik-Support:

Bei User-Registrierung rechtzeitige Update-/Upgrade-Information und Support-Unterstützung.

Senden Sie uns bitte Ihre Registrierungskarte.



ELECTRONIC ARTS

DELUXE: Software für den Amiga.

Markt&Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel, in Computer-Fachgeschäften oder bei Ihrem Buchhändler.

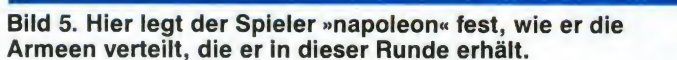
Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Fragen Sie Ihren Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 41 56 56. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5 87 13 93-0; Rudolf Lechner & Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 67 75 26; Ueberreuter Media Verlagsges.m.bH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 48 15 43-0.

Haben Sie eine Funktion, zum Beispiel »Angriff«, versehentlich gewählt, können Sie das Feld »Abbruch« (Mitte unten) anklicken. Wählen Sie dann die nächste Operation. Der Abbruch ist beispielsweise auch dann notwendig, wenn Sie ein Land für einen Angriff ausgewählt haben, das nur



Ein Feldherr sollte möglichst gut mit Karten umgehen können. Das gilt auch für unser Strategiespiel. Wurde mindestens ein Land in einer Spielrunde erobert, gibt es eine der wertvollen Karten. Die Karten tragen die Symbole Soldat, Reiter und Kanone (Bild 2). Besitzt ein Spieler drei Karten mit dem gleichen Symbol oder eine Serie aller Symbole, dann kann er diese gegen Armeen eintauschen. Der erste Spieler, dem es gelingt, sich durch mehrfachen Tauschen die Anzahl der Bonusarmeen schnell vergrößern zu lassen, ist am besten angestiegen. Umwälzende Veränderungen sind mit dem Einsatz der Zusatzarmeen schnell herbeigeführt.

Aus den genannten Gründen sollten Sie unbedingt versuchen, in jeder Spielrunde mindestens ein Land zu erobern.

Ein erfolgreicher Feldherr schnappt sich so schnell wie möglich einen der kleinen Kommandanten.

Bei mehreren Mitspielern sind Bündnisse eine tolle Sache. Kämpfen Sie gemeinsam gegen einen Gegner, bewahren Sie den Frieden an gemeinsamen Grenzen. Wenn Sie von einem »Bündnispartner« überfallen werden, nehmen Sie das nicht persönlich: Es ist doch nur ein Spiel!

**Fortsetzung
des Artikels
auf Seite 134**

Helfen Sie mit, Ihr Amiga-Sonderheft noch besser zu gestalten. Wenn Sie diesen Fragebogen ausfüllen und an unserer Leserumfrage teilnehmen, haben Sie zudem die Chance, ein Overdrive-Doppellaufwerk für Ihren Amiga zu gewinnen.

Bevor Sie sich auf den Fragebogen stürzen, sollten Sie einige Dinge wissen:

- Ihre Antworten, ob Lob oder Kritik, haben keinen Einfluß auf Ihre Gewinnchancen
- Bei fast allen Fragen sind Mehrfachnennungen möglich
- Fragen, bei denen freie Textangaben vorgesehen sind (Typ des Druckers etc.), tragen Sie bitte eine möglichst genaue Typenbezeichnung ein. Zum Beispiel: »Drucker NEC P6 C«.
- Sie müssen Ihr Heft nicht zerschneiden, eine Fotokopie der Umfrage-Seiten genügt auch.
- Wenn Sie an der Verlosung des Overdrive-Systems teilnehmen wollen: Vergessen Sie Ihre Adresse nicht.



Einsendeschluß ist der 31.12.1988 (Datum des Poststempels).
Der Rechtsweg ist ausgeschlossen. Mitarbeiter der Markt &
Technik Verlag AG sowie deren Angehörige dürfen an der Um-
frage nicht teilnehmen.

1. Welche Themenbereiche sollten in den Amiga-Sonderheften angesprochen werden.

(Bewerten Sie die Wichtigkeit mit einer Note zwischen 1 und 6)

Anwendungslistings	___
Bauanleitungen	___
Betriebssysteme	___
Bücher	___
CLI	___
DFÜ	___
Digitalisieren/Scannen	___
Drucker-Tests	___
Einsteigerthemen	___
Festplatten/Floppylaufwerke	___
Grafik	___
Grafik-Animation	___
Grundlagen	___
Hardware-Tests	___
Intuition	___
Kaufmänn. Anwendungen	___
Messen/Steuern/Regeln	___
Musik/Midi	___
Problemlösungen	___
Programmieren	___
Programmierkurse	___
- Assembler	___
- C	___
- Basic	___
- Pascal	___
- Modula	___
Programmiersprachen allgemein	___
Schule/Ausbildung	___
Software-Tests zu einem bestimmten Thema	___
Spiele-Listings	___
Stories	___
Techn./Wiss. Anwendungen	___
Themen für Profis	___
Tips & Tricks	___
Video-Anwendungen	___
Workbench	___

2. Wie ist Ihre Meinung zum Amiga-Sonderheft

Bewerten Sie die einzelnen Punkte mit einer Note zwischen 1 und 6)

Informativ	___
Aktuell	___
Kritisch	___
Sachlich	___
Verständlich	___
Ansprechend	___
Preiswert	___
Unverzichtbar	___

3. Besitzen Sie einen Amiga?

Nein ☐ Amiga 500 ☐ Amiga 1000 ☐ Amiga 2000 ☐

4. Wollen Sie demnächst einen Amiga kaufen?

Nein ☐ Amiga 500 ☐ Amiga 1000 ☐
Amiga 2000 ☐ Amiga 2500 UX ☐ Amiga 2500 AT ☐

5. Besaßen Sie oder besitzen Sie einen anderen Computer?

Nein ☐ Ja ☐

Wenn ja, welchen?

	Wird noch benutzt	Besitze ich	Besaß ich
C 64	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
CPC	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Atari XL/XE	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Atari ST	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PC	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Verkauft am: _____

Andere: _____

6. Wie lange besitzen Sie schon Ihren Amiga?

Weniger als drei Monate	<input type="radio"/>
Bis sechs Monate	<input type="radio"/>
Bis ein Jahr	<input type="radio"/>
Über ein Jahr	<input type="radio"/>
Über zwei Jahre	<input type="radio"/>

7. Wie ist Ihr Amiga ausgestattet?

a) Monitor

Spezieller Amiga-Monitor	<input type="radio"/>	Fernsehgerät	<input type="radio"/>
Multisync	<input type="radio"/>		

b) Massenspeicher

Diskettenlaufwerke	<input type="radio"/>	Anzahl	___
Festplatte	<input type="radio"/>	Anzahl	___ Kapazität ___ MByte
Sidecar	<input type="radio"/>		
PC-Karte (A 2000)	<input type="radio"/>		

c) Speichererweiterungen

keine	<input type="radio"/>	512 KByte	<input type="radio"/>
1 MByte	<input type="radio"/>	2 MByte	<input type="radio"/>
3 MByte	<input type="radio"/>	Mehr als 3 MByte	<input type="radio"/>

d) Drucker

Keinen	<input type="radio"/>	Matrix	<input type="radio"/>	Laser	<input type="radio"/>
Tintenstrahl	<input type="radio"/>	Thermo	<input type="radio"/>		

Typ: _____

Farbdrucker:	Ja	<input type="radio"/>	Nein	<input type="radio"/>
--------------	----	-----------------------	------	-----------------------

e) DFÜ

Akustik-Koppler	<input type="radio"/>	Modem	<input type="radio"/>
-----------------	-----------------------	-------	-----------------------

f) Digitizer/Scanner/Genlock etc.

Typ: _____

8. Wie wird Ihr Amiga von Ihnen vorwiegend genutzt?

Im privaten Bereich:

Programmieren	<input type="radio"/>	Anwendungen	<input type="radio"/>
Spielen	<input type="radio"/>		

In der Ausbildung:

Schule	<input type="radio"/>	Fortbildung	<input type="radio"/>
Studium	<input type="radio"/>		

Im Beruf:

Hauptberuflich:	<input type="radio"/>
Nebenbeschäftigung	<input type="radio"/>

9. Was machen Sie mit Ihrem Amiga

	intensiv	gelegentlich	nie	würde mich interessieren
Animation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Dateiverwaltung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
DFÜ	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Desktop Publishing	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Digitalisieren/Scannen	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Elektronik/Basteln	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Grafik/Malen	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Kaufm. Anwendung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lernprogramme	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Messen, Steuern, Regeln	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Musik/Midi	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Programmieren	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Spielen	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Textverarbeitung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sonstiges				

10. Wie stufen Sie sich ein?

Einsteiger ohne Vorkenntnisse ☐ Einsteiger ☐ Fortgeschrittener ☐
 Erfahrener Fortgeschrittener ☐ Profi ☐ Ist mein Beruf ☐

11. Was machen Sie mit den Listings im Amiga-Sonderheft?

Tippe alle ein ☐
 Tippe nur die kurzen ein ☐
 Interessiere mich nur für die Struktur ☐
 Interessieren mich nicht ☐
 Besorge mir sie auf Diskette von Freunden ☐
 Nehme den Programmservice in Anspruch ☐

12. Wie groß sollte Ihrer Meinung nach der Listing-Teil im Amiga-Sonderheft sein?

ca. 20 Seiten ☐ ca. 40 Seiten ☐
 ca. 50 Seiten ☐ mehr als 50 Seiten ☐

13. Wie viele Listings sollten in einem Amiga-Sonderheft abgedruckt werden?

Bis 5 Listings ☐ Bis 10 Listings ☐
 Bis 15 Listings ☐ Mehr als 15 Listings ☐

14. Wie stehen Sie zu der Möglichkeit, Programme nicht mehr im Heft abzudrucken und diese nur noch auf Diskette anzubieten. Die Information und die Zahl der Programme im Amiga-Sonderheft könnte so wesentlich höher sein

Alle Programme nur auf Diskette ☐
 Nur besonders lange Programme nur auf Diskette ☐
 Alle Programme auch im Amiga-Sonderheft als Listing ☐

15. Welche Programmiersprachen kennen Sie, interessieren Sie?

Sprache	kenne ich	besitze ich	beherrsche ich	würde ich gerne im Heft haben	wünsche ich mir einen Kurs
Assembler	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
ADA	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Basic	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Forth	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Logo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lisp	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Modula	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Pascal	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prolog	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Andere					

16. Fragen zur Person

a) Alter: _____ Jahre
 b) Geschlecht männlich ☐ weiblich ☐

c) Welche Hobbies haben Sie?

Basteln ☐
 Computer ☐
 Elektronik ☐
 Freizeit-Verein ☐
 Sport ☐
 Lesen ☐
 Sonstige ☐

d) Schulabschluß, Ausbildung, bzw. angestrebter Abschluß

Hauptschule ☐ Mittlere Reife ☐ Lehre ☐
 Fachoberschule ☐ Abitur ☐
 Fachhochschule ☐ Universität ☐

e) Anschrift

(Ausfüllen, wenn Sie an der Verlosung teilnehmen wollen)

Name: _____

Straße: _____

Ort: _____

Telefon: _____

Ich bin damit einverstanden, daß meine hier gemachten Angaben elektronisch ausgewertet werden.

Unterschrift _____

Profilauflwerk 3,5"

Metallgehäuse • einstellbare Laufwerksnummer mit Displayanzeige • digitale Trackanzeige • Write Protect am Laufwerk schaltbar • abschaltbar • durchgeschleif-ter Bus
1 Jahr Garantie
Super ALCOMPPreis

329,-

Laufwerk 5,25"

40/80 Track • Laufwerksbus durchge-
schleift • abschaltbar • einstellbare
Adressen • MS-DOS-kompatibel • mit
Diskchange

Super ALCOMPPreis

298,-

HD 1,6 MB (umschaltbar)

318,-

Amigafarbene Blende

+10,-

Write Protect Schalter

+15,-

Gemischtes Doppel 3,5/5,25"

einzel ein-/abschaltbar • einstellbare
Laufwerksnummern mit Anzeige • durch-
geschleif-ter Bus • bei 5,25" 40/80 Tracks
umschaltbar • Metallgehäuse • 1 Jahr Ga-
rantie

Super ALCOMPPreis

548,-

**ausgereifte Ingenieurlei-
stung • 14 Tage
Umtauschrecht • fast
alle IC's gesockelt • nur
professionelle Leiter-
platten • Bauteile
namhafter Hersteller •
mit Bedienungsanleitung**

3,5" Laufwerk

Für alle Amiga's • einstellbare Gerätenum-
mer • abschaltbar • Metallgehäuse • su-
perflach • 1 Zoll (2,54cm) • durchge-
schleif-ter Bus • TEAC Laufwerk

1 Jahr Garantie

komplett anschlussfertig

239,-

Amigafarbene Blende

+10,-

Basistaufwerke

1 Jahr Garantie

TEAC FD 135 FN 3,5" 1MB superslimline

218,-

TEAC FD 55 GFR 5,25" 40/80 Tracks

239,-

Amigafarbene Blende

+10,-

1,6 MB Diskchange

259,-

3,5" Gehäuse

25,-

5,25" Gehäuse

25,-

Gehäuse für "Gemischtes Doppel"

65,-

Bootselector

19,90

Amiga Eprommer

• Für A 500/1000

• Expansionsportsanschluß

• Für EPROM's 2764-27011 (8K-128K)

Alle A-Typen und CMOS-Typen

• Funktionen:

LEERTEST

LADEN VON DISK

VERGLEICHEN

SPEICHERN AUS DISK

AUSLESEN

HEXDUMP

BRENNEN

• vier Programmieralgorithmen

50mS/Byte - Superschnell 64K-1,5 min

• Programm zum Generieren und Brennen

von Kickstarts direkt von Diskette oder

aus ROM

• Mit Software + Gehäuse

225,-

Meß- und Steuerinterface

• 8 ADC-Kanäle 0-2,55V in 0,01V Stufe

• 1 DAC-Kanäle 0-2,55V in 0,01V Stufe

Genauigkeit: 1,5 LSB

• 8 frei programmierbare TTL-I/O Kanäle

• Mit Gehäuse, Anschlüsse auf Schraub-

klemmen

• interne Referenzspannung

• Expansionsanschluß

• Einfache Programmierung in Basic mög-

lich Multitasking tauglich

incl. DEMO-Software auf 3,5" Diskette

239,-

500er Speichererweiterung

Für 512k zusätzliches RAM • alle RAM-s
gesockelt • selbstkonfigurierend • ab-
schaltbar • Uhrenschtaltung auf Platine mit
Akku- bzw. Batteriepufferung nachrüstbar

Komplett mit 512k

Preis auf Anfrage

Superpreis mit Uhr

Preis auf Anfrage

Bauteilesatz für Uhr ohne Akku

24,-

Leerplatine mit Stecker

*39,-

*mit Schaltplan und Bestückungsliste

Laufwerkanschlußkabel

Zum Anschluß von Laufwerken an alle Amigas •

mit Ansteuerlektronik

Für 3,5" Laufwerk

39,-

Für 5,25" Laufwerk

49,-

Steckplatzerweiterung

3-fach für Laufwerke

Jeder Steckplatz abschaltbar und einstellbare
Laufwerksnummer • Steckplatzweiterung di-
rekt am Amigaehäuse • Dadurch keine Kabel-
längenprobleme

Anschlußfertig zum Super ALCOMPPreis

49,-

Soundsampler

Für alle Amiga's mit Software • Typa bei
Bestellung bitte angeben • 8-Bit Daten-
breite • Betrieb am Parallelport (Drucker-
port) • Mit Vorverstärker für Micro-An-
schluß (Cinch-Buchsen) • Musik- und
Sprachdigitalisierung möglich • Arbeitet
mit fast allen Digitizer-Programmen •
Formschönes Gehäuse
Super ALCOMPPreis

79,-

Sampler Studio

• Professionelles Sampler-Programm • 4-Kanal-
Technik • speichern auf 4 Disketten hinterinan-
der möglich • alle gängigen Formate (IFF, Data,
Future) • Echtzeitdisplay mit Zoomfunktion •
viele Verformungsmöglichkeiten • Echo, Hall,
Reverse

69,-

Paket: Sampler + Software

129,-

MIDI-Interface

4 Kanäle einschließlich 1 Thru • Optische
Datenanzeige • Formschönes Gehäuse
Wahnsinnspreis von nur

89,-



Selbstbootende Harddisk für Amiga ohne PC-Karte!

Die Amiga-Festplatte von ALCOMP:

• Selbstbootend wie "Card" oder "Rad"! • Als Einbau-Festplatte für den "Amiga 2000" • Als Exter-

ne Einheit für den "Amiga 500" und 1000 mit Gehäuse, eigenem Netzteil und Erweiterungsanschluß

• Erhältlich mit 20, 30, 40 und 65 Megabyte • Kopiert 1 Megabyte in unter 4 Sekunden • Speichert

schneller als "1,2-Ramdisk" • Läuft mit "FastFileSystem" • Einfach einstecken, Formatieren,

"Mountlist" und "Startup-Sequence" ändern und los geht's!

Entwickler: Stephan und Stefan

Für den Selbstbau: Harddisk-Interface incl. Steuersoftware • Anschluß mit Slot für Omti-Controller



Kickstartumschaltung

Bauen Sie die anderen Kickstart-Versionen in Ihren
Amiga 500 • Einfacher Einbau ohne Löten • für
Original-Kickstart-ROM und 2 zusätzliche Versio-
nen auf EPROM • EPROM-Programmierservice auf
Anfrage

SuperALCOMPPreis

59,-

Kickstartversion auf EPROM's

120,-

Userport + Experimentierkarte für Expansionport

Mit Lochraster und 2 x 6522 Ports

Leer

59,-

komplett aufgebaut

89,-

**Wir suchen ständig Hardware-Ent-
wicklungen. Wir garantieren gute
Umsatzprovisionen und ehrliche
Abrechnung**

kostenloses Info anfordern!!!

Bestellung und Versand

ALCOMP

A. Lanfermann

Lessing Str. 46

5012 Bedburg

Tel. 0 22 72/15 80

Nachnahmeversand NN-Spesen 7.50

DM b. Vorkasse 3.- DM. Auslandsbe-

stellungen: Nachnahmeversand NN-

Spesen 10.- DM b. Vorkasse 5.-DM.

Wir liefern Ihnen auf Ihre Rechnung

und Gefahr zu den Verkaufs- und Liefer-

bedingungen des Elektronikgewerbes.

Postgiroamt Köln

(BLZ 370 100 50) 275 54-509

Finanzen im Griff

**Überlassen Sie nichts dem Zufall.
Wenn es um Ihr Geld geht, genügt kein gelegentlicher
Blick auf den Bankauszug von letzter Woche.
Mit dem elektronischen Haushaltsbuch haben Sie Ihre
Kasse stets unter Kontrolle.**

Mit dem »Haushaltsbuch« für den Amiga wird die Überwachung Ihrer täglichen, monatlichen oder jährlichen Einnahmen und Ausgaben zum Kinderspiel. Die Bedienung des Programms ist schnell erlernt, die Menüs sind mit der Maus aufzurufen. Lediglich die Eingabe der Kontennamen und der Geldbeträge erfolgt über die Tastatur.

Haben Sie erst einmal ein Konto angelegt, die ersten Beträge verbucht und den Stand auf Diskette gespeichert, genügt ein Mausklick, um eine Monats- oder Jahresübersicht auszugeben. Weitere Menüs wie DISK, ZEIT, SYSTEM, AUSGABE und DRUCKER geben dem Programm einen professionellen Touch.

So führen Sie Ihre Konten

Der Arbeitsbildschirm des Basic-Programms »HAUSHALTSBUCH« (Listing 1) unterteilt sich in zwei Windows:

1. Das Eingabe- und Ausgabefeld (oben).

2. Aktuelle Informationen wie Uhrzeit, Speicherplatz und Programmhilfen.

Vor dem ersten Start des Programms sind einige Vorarbeiten nötig. Beachten Sie bitte den Kasten mit den Hinweisen zu diesen Punkten. Besitzer der Programmservice-Diskette können allerdings sofort mit der Verwaltung beginnen.

Die Programmdiskette muß sich im Laufwerk df0: be-

finden. Basic-Programmierer können das Programm bei Bedarf jedoch leicht umschreiben, so daß ein zweites Laufwerk angesprochen wird.

Menü à la carte

Das Haushaltsbuch stellt zahlreiche Funktionen zur Verfügung, die bei gedrückter rechter Maustaste (Menütaste) angewählt werden.

Bevor Sie mit der Eingabe von Daten beginnen, ist ein neues Konto anzulegen. Insgesamt können zehn Konten in einer Datei geführt werden, was für die private Anwendung sicher ausreicht.

Wählen Sie den Punkt »neues Konto« im Menü »Option« (Bild 1). Geben Sie anschließend den Namen, beispielsweise »Gehalt«, ein. Speichern Sie diesen auf Diskette, indem Sie die folgende Frage mit »Ja« beantworten (mit der Maus anklicken).

Wenn Sie anschließend im »Verzeichnis« nachschauen, finden Sie dort den Eintrag des angelegten Kontos.

Die erste Buchung

Sie können nun weitere Konten eröffnen, oder aber in das eröffnete Konto Einträge buchen. Wählen Sie dazu mit der Maus im »Verzeichnis« das Konto, in das Einträge gebucht werden sollen. Sie erkennen die erfolgreiche Auswahl daran, daß im Ausgabefeld unten links der Kontoname erscheint.

Öffnen Sie dann das Menü »Option« und wählen »Buchen«. Bestimmen Sie anschließend, ob eine Ausgabe oder eine Einnahme verbucht werden soll. Sie werden dann aufgefordert, eine Bezeichnung für den Buchungsbetrag einzutippen. Schließen Sie die Eingabe mit <RETURN> ab. Danach ist die Höhe des Betrags einzugeben. Hier sind nur Zahlen und, als Trennung für die Nachkommastelle, der Punkt erlaubt.

Sollten Sie versehentlich ein ungültiges Zeichen eingegeben haben, erscheint die Meldung »?Redo from start«. Drücken Sie dann <RETURN> und buchen erneut.

Die in der Arbeitssitzung bisher erfolgten Buchungen sehen Sie mit der Wahl des Me-

nüpunktes »Buchungen«.

Haben Sie auf die beschriebene Art alle Buchungen vorgenommen, speichern Sie diese mit dem Punkt »Eintragen« auf Diskette. Erst dann sind Ihre Daten endgültig festgehalten. Bei längeren Sitzungen sollten Sie zwischendurch immer wieder auf Diskette speichern, um Datenverlust, beispielsweise durch einen Stromausfall, zu vermeiden.

Sie können die auf Diskette gespeicherten Daten in die-



ANWENDUNGEN



Einzelkonto		LEBENSMITTEL		Jahr 1988 Monat 09		Maustaste	
Tag	Monat	Jahr	Bezeichnung	Betrag			
16	09	1988	GROBEINKAUF	-455.78			
16	09	1988	BROTCHEN	-2.80			
16	09	1988	FLEISCH	-21.70			
16	09	1988	SÜSSIGKEITEN	-4.55			
				Saldo:		-484.83	

Zeit	12:23:28	Arbeitsdauer	h: 0 n: 0 s: 0
Datum	16-09-1988	Buchungen	0
Bezugszeit	16-09-1988	Speicher	553632 4147 25722
Kontoname	LEBENSMITTEL		

von A.N. M. M.F.		Hilfsverwaltung		Maustaste	
Monatskonto		Jahr 1988 Monat 09			
Konto	Betrag				
GEHALT	15100.55				
AUTO	0.00				
HOBBYS	0.00				
LEBENSMITTEL	-404.83				
Saldo:		14615.72			
Zeit	12:42:14	Arbeitsdauer	h: 0	n: 0	s: 0
Datum	16-09-1988	Buchungen	0		
Bezugszeit	16-09-1988	Speicher	556200	4407	26234
Kontoname	LEBENSMITTEL				

sicht Monat« und »Übersicht Jahr« zeigen alle gebuchten Einträge in dem betreffenden

Das Programm »Haushaltsbuch« und der zugehörige Datenspeicher benötigen sehr

20



Stoff für Ihren Amiga



Markt&Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel, in Computer-Fachgeschäften oder bei Ihrem Buchhändler.

Markt & Technik
Zeitschriften · Bücher
Software · Schulung

Fragen Sie Ihren Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 41 56 56. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5 87 13 93-0; Rudolf Lechner & Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 67 75 26; Ueberreuter Media Verlagsges.m.b.H (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 48 15 43-0.

ANWENDUNGEN

viel Speicherplatz, so daß mindestens eine 512 KByte Speichererweiterung benötigt wird. Ist zusätzlich eine batteriegepufferte Echtzeituhr vorhanden, wird diese vom Programm

unterstützt. Sollte die Uhr nicht vorhanden sein, müssen Datum und Uhrzeit vor dem Laden eingestellt werden.

(M. Fernholz/Walter Kurt/rs)

Bild 4. Verändern Sie die Bezugszeit, um Buchungen ein anderes Datum als das Systemdatum zuzuweisen

Bild 5. Sogar die Farben des Arbeitsbildschirmes sind zu verändern

Setzen Sie bitte vor dem Start des Programms die Systemzeit, falls diese nicht mit dem aktuellen Datum und der Uhrzeit übereinstimmt. Diese Einstellungen führen Sie bitte mit der Workbench durch. Fortgeschrittene setzen die Zeit natürlich vom CLI aus..

Das Programm erwartet im Unterverzeichnis »SCHUBLADE« die vom Anwender eingestellte Farbtabelle. Da diese vor dem ersten Start des Programms nicht vorhanden sind, gehen Sie bitte folgendermaßen vor:

Legen Sie auf der Diskette mit dem Programm »Haushaltsbuch« das Subdirectory »SCHUBLADE« an. Dazu kopieren Sie das Verzeichnis »EMPTY« von der Workbench und benennen dies anschließend mit dem Rename-Befehl in »SCHUBLADE« um.

Legen Sie die Diskette in das Laufwerk df0:, laden Sie anschließend Amiga-Basic und dann das »Haushaltsbuch« (»OPEN« mit der rechten Maustaste anwählen). Geben Sie bitte im Basic-Fenster (links) nach Anklicken mit der linken Maustaste folgende Zeilen ein:

```
chdir "df0:schublade"
open "o", #1, "Zeit"
print #1, "195501"
close
```

Beachten Sie bitte unbedingt die korrekte Groß-/Kleinschreibung! Schreiben Sie nun im List-Fenster (rechts) vor die Zeile:

OPEN "A", #1, "Farben ...

die Buchstaben »REM«. Sie müssen den Mauszeiger dazu an den Anfang der Zeile bewegen und die linke Maustaste drücken. Ein senkrechter Strich erscheint in der Zeile. Geben Sie nun die Buchstaben »REM« ein, drücken einmal die Space-Taste und bewegen den Mauszeiger in die nächste Zeile. Starten Sie anschließend das Programm (»Start« anklicken). Mit der Option »Farben« im Menü »System« stellen Sie nun die Farben Ihrer Wahl ein. Wenn Sie die Farben nicht verändern wollen, klicken Sie bitte alle acht Farben der Reihe nach an und anschließend das Feld »Fertig«. Erst dann sind die Farbwerte zum Speichern vorbereitet!

Nach dem Einstellen fragt das Programm, ob Sie die Farben speichern wollen. Geben Sie unbedingt »JA« an. Das File »Farben« wird nun erzeugt. Beenden Sie anschließend die Arbeit mit dem Haushaltsbuch, erscheint die Meldung »Current program is not saved. Do you want to save it before proceeding?«. Klicken Sie mit der linken Maustaste bitte »NO« an. Nun kann das »Haushaltsbuch« ohne Schwierigkeiten gestartet werden.

Programmname:	Haushaltsbuch
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2
Bemerkung:	Bitte vor Programmstart im Unterverzeichnis »Schublade« die Dateien »Zeit« und »Farben« anlegen

```
1 Xd0 start:
2 Lc REM +++Haushaltsverwaltung von Andreas Noch und Martin Fernh
   olz +++
3 gS REM +++Erstellungsjahr Januar bis Februar 1988 +++
4 He REM *** Bitte nur in einem Monat Buchen (EINTRAGEN!!!)
5 QO REM *** Das Erstellen eines neuen Buchungsjahres dauert ca
   .1.5 min (Schreiben der Jahresdateien)
6 NS REM *** Die Schublade MUß beinhalten die Dateien Zeit und
   Farben !!!!
7 ZB REM *** Das Programm arbeitet fehlerfrei mit 1 Mbyte Speic
   her !
8 L3 REM *** Das Programm arbeitet auch mit 2 Laufwerken (Änder
   ung auf 2 Laufwerke CHDIRdf0:Schublade Befehle im Program
   m auf DF1 verändern)
9 zd REM *** Die niedrigste Bezugszeit ist 1988
10 GJ CHDIR "df0:Schublade"
11 mK CLEAR ,50000&
12 oD OPTION BASE 1
13 5U DEFDBL a-z
```

```
14 XE index=0
15 n5 DIM verzeichnis$(19),flag$(19),nummer(99),konto$(99),jahr$(
   99),monat$(99),tag$(99),buchung$(99),betrag$(99)
16 KS GOSUB systemzeit
17 9o GOSUB bildschirm
18 Mh GOSUB einlesen
19 11 OPEN "I", #1, "Farben":FOR i=0 TO 7:INPUT #1,rot(i+1),gruen
   (i+1),blau(i+1):PALETTE i,rot(i+1),gruen(i+1),blau(i+1):NEXT
   i:CLOSE #1
```

Listing 1. Das Haushaltsbuch bringt Ordnung in Ihre Finanzen. Bitte mit dem Checksummer (Seite 159) eingeben.

Fortsetzung des Listings auf Seite 143

ANWENDUNGEN

Intelligenter Sprachtrainer

Ohne ausreichenden Wortschatz ist keine Fremdsprache zu beherrschen. Erweitern Sie Ihr Vokabular einer beliebigen Sprache mit diesem großartigen Programm. Mit dem Vokabel-Trainer sind nicht nur exotische Sonderzeichen darzustellen, er reagiert auch auf vorhandene Lücken in Ihrem Sprachschatz.

Die Fähigkeiten des »Vokabel-Trainers« (Listing 1 und 2) gehen weit über die Möglichkeiten vergleichbarer Programme hinaus. Eine Übersicht der hilfreichen Funktionen wird Sie überzeugen:

- mausunterstützte Benutzeroberfläche
- nationale Sonderzeichen auf Funktionstasten
- Suchfunktion für jede Vokabel oder das passende Synonym im Speicher und auf Diskette
- Vokabelabfrage nach drei verschiedenen Kriterien (Fremdwort, Übersetzung und Schreibweise)
- durch Statistikfunktion für jede einzelne Vokabel ständiger Überblick des eigenen Wissensstandes
- Datumsspeicherung der Abfrage

Fragen Sie sich, warum Vokabeln mit Hilfe eines Computers besser und schneller gelernt werden? Computer-unterstütztes Lernen bietet eine Reihe von Vorteilen. Einerseits eignen Sie sich beim Eintippen wesentlich besser die Schreibweise der Vokabeln an. Andererseits lernen Sie die tatsächliche Bedeutungen ein-

zelner Vokabeln und nicht eine Abfolge verschiedener Wörter. Beim üblichen Lernvorgang mit Hilfe eines Lehrbuches prägen sich die Vokabeln primär in der Reihenfolge der Lektion ein, weniger die Bedeutungen der einzelnen Vokabeln. Dies führt häufig zu dem Irrglauben, die Vokabeln seien schon bestens bekannt, obwohl bei einem aus dem Zusammenhang gerissenen Wort Unsicherheit auftritt.

Der Vokabeltrainer bietet weiterhin den großen Vorteil, Vokabeln dem Wissensstand des Benutzers angepaßt abzufragen. Dies geschieht durch ein automatisches Auswahlverfahren. Vokabeln, die Sie weniger gut beherrschen, werden häufiger abgefragt als bekannte Worte.

Die Möglichkeiten des Vokabeltrainers sind damit noch lange nicht erschöpft. Mehrere Anwender können ihre Vokabeln auf einer Diskette getrennt voneinander verwalten. Die Statistikfunktion erlaubt eine genaue Auswertung jeder einzelnen Vokabel, jeweils aufgeteilt auf die drei Abfragearten (siehe unten). Zusätzlich werden Datum und Uhrzeit mit auf Diskette gespeichert. Da-

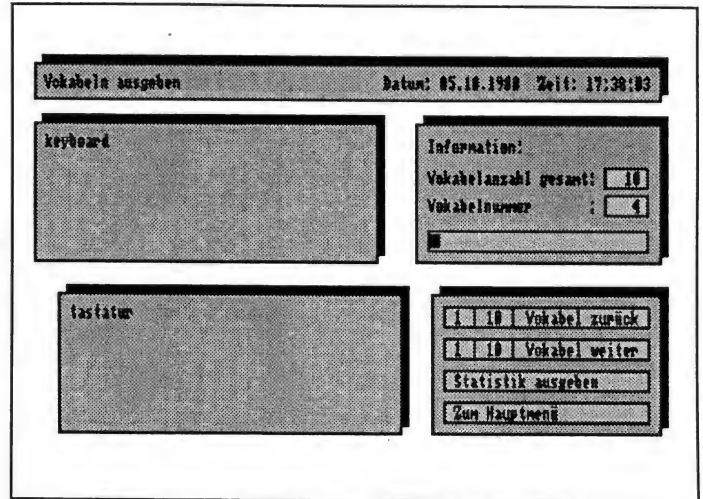


Bild 3. Die Maske für die Eingabe neuer Vokabeln

mit ist genau nachzuvollziehen, wann welche Vokabeln zuletzt abgefragt wurden und ob eventuell eine Wiederholung der entsprechenden Lektion nötig ist.

Die Vorbereitung

Im folgenden möchte ich zunächst einige Hinweise zum Abtippen geben und danach auf die einzelnen Funktionen des Programms ausführlich eingehen.

Der Vokabeltrainer ist vollständig in Amiga-Basic auf einem Amiga 2000 programmiert worden. Das Programm ist für »Basic-Kundige« leicht eigenen Wünschen anzupassen. Der Trainer läuft problemlos auf einem Amiga 500 oder Amiga 1000 mit einem Hauptspeicher von mindestens 512 KByte.

Vor der Eingabe des Hauptprogramms muß in jedem Fall der von Amiga-Basic zur Verfügung gestellte freie Speicherplatz von 25000 Byte auf mindestens 70000 Byte vergrößert werden. Für Einzeldateien mit mehr als 200 Vokabeln sollte er sogar etwa 100000 Byte betragen.

Das Hauptprogramm ist länger als 25000 Byte. Damit bei einem neu gestarteten Computer keine Schwierigkeiten auftreten, ist zunächst ein kleines Ladeprogramm mit Namen »Trainer« (Listing 1) zu laden und zu starten. Dieses Programm vergrößert den freien Speicherplatz und ruft das Hauptprogramm auf. Damit dieses Ladeprogramm das Hauptprogramm auf Diskette

findet, speichern Sie den Hauptteil unter dem Dateinamen »Trainer HP« ab. Beachten Sie bitte, daß die Diskette sich im Laufwerk befindet, das im Basic-Lader angegeben ist (z.B. df0:).

Die maximal mögliche Anzahl von Vokabeln in einer Datei läßt sich leicht verändern, indem Sie die Variable »mmv« am Anfang des Programms Ihren Bedürfnissen anpassen. Ferner ist zu beachten, daß auch der im Ladeprogramm enthaltene Befehl »CLEAR« an den größeren Speicherbedarf angepaßt werden muß (durch Veränderung der Zahl hinter dem CLEAR-Befehl).

Jetzt geht's los

Nach dem Programmstart werden Sie zunächst aufgefordert, Ihren Namen einzugeben. Der Name darf maximal 20 Zeichen beinhalten. Sollte ein File mit dem gewählten Namen noch nicht auf Diskette vorhanden sein, legt das Programm dieses File an. Ist die Namensdatei schon auf der Diskette vorhanden, lädt der Computer alle Daten, die mit diesem Namen zusammenhängen. Hierzu gehören die Namen der Vokabeldateien auf der eingelegten Diskette, die vom Anwender eingestellten Grundfarben sowie die Benutzerdaten mit Datum und Uhrzeit. Mit Hilfe des Anwendernamens ist es möglich, eine eigene, von anderen Daten getrennte, Vokabelbibliothek aufzubauen. Diese Funktion ist interessant, wenn auf Diskette

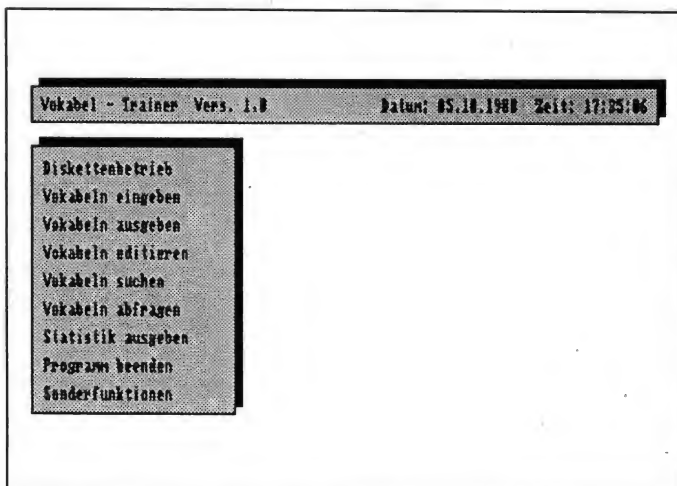


Bild 1. Das Hauptmenü des Vokabeltrainers

ANWENDUNGEN

Bild 2. Ein Beispiel für die Vokabelausgabe

nach einer bestimmten Vokabel gesucht wird.

Nach der Namenseingabe erscheint das Hauptmenü (Bild 1) mit Datum und Uhrzeit (rechts oben auf dem Bildschirm). Ist eine Hardware-Uhr vorhanden, übernimmt das Programm die dort abgelegten Werte. Anwender ohne Hardware-Uhr sollten das aktuelle Datum und die Uhrzeit vor Aufruf des Programms entweder über die »Preferences« oder mit Hilfe des DATE-Befehls eingeben.

Hilfreiche Maus

Das gesamte Programm ist speziell auf Maussteuerung ausgelegt, die Tastatur wird ausschließlich für Texteingaben verwendet.

Diskettenbetrieb

Der erste Punkt im Hauptmenü unterteilt sich in folgenden Funktionen:

1. Vokabeldatei laden

Um eine Datei von Diskette in den Arbeitsspeicher des Computers zu laden, klicken Sie den gewünschten Dateinamen an. Sollten nicht alle gespeicherten Dateien im Ausgabefenster Platz finden, verschieben Sie mit den beiden Kästchen links unten den sichtbaren Ausschnitt.

2. Vokabeldatei speichern

Wurde eine Datei erstellt oder ergänzt, kann sie mit dieser Funktion auf Diskette gespei-

chert werden. Haben Sie eine neue Datei erstellt, verlangt das Programm einen Dateinamen von maximal 20 Zeichen. Soll eine vorher geladene Datei wieder abgespeichert werden, erscheint im Namensfeld die alte Bezeichnung der Datei. Mit <RETURN> übernehmen Sie diesen Namen. Wollen Sie diese Funktion abbrechen, klicken Sie das Feld »Zurück zum Hauptmenü« an. Diesen Abbruch zum Hauptmenü finden Sie bei vielen Programmfunktionen.

3. Vokabeldatei löschen

Nach Anklicken dieses Menüpunktes wird eine nicht mehr benötigte Datei von Diskette gelöscht.

4. Diskette wechseln

Sollten Sie eine neue Diskette mit Vokabeln einlegen, ist dringend erforderlich, diese Funktion anzuwählen, da das Betriebssystem ansonsten die alte Diskette zurückverlangt. Der Vokabeltrainer liest die Dateinamen erst dann von der neu eingelegten Diskette.

Vokabeln eingeben

Diese Funktion (Bild 3) erlaubt Ihnen einerseits, eine neue Vokabeldatei zu beginnen, andererseits eine schon vorhandene Datei mit neuen Vokabeln zu ergänzen. Nach Auswahl der Funktion erscheinen in der oberen Hälfte des Bildschirms zwei Eingabefelder. Hier tippen Sie das Fremdwort und die Übersetzung ein, jeweils mit anschließendem

<RETURN>. Links unten sehen Sie die Belegung der Funktionstasten mit den nationalen Sonderzeichen. Voreingestellt sind hier die französischen Sonderbuchstaben. Alle vom Amiga unterstützten nationalen Zeichensätze sind auf der Workbench im Directory »DEVS/KEYMAPS« abgelegt. Wollen Sie einen anderen Zeichensatz verwenden, müssen Sie die DATA-Zeilen am Ende des Hauptprogramms verändern. Die ASCII-Codes der Sonderzeichen entnehmen Sie bitte dem Amiga-Handbuch.

Die Anzeige rechts unten zeigt Informationen über die Anzahl der Vokabeln im Speicher und die Speicherbelegung.

Verschiedene Bedeutungen einer Vokabel (Synonyme) sollten mit Kommata getrennt werden. Beispielsweise bedeuten die beiden englischen Worte »Hit« und »Beat« zu deutsch »Schlagen«. Bei der Abfrage berücksichtigt das Programm diese Synonyme. Ins Hauptmenü zurück kommen Sie mit <RETURN>, ohne eine Vokabel einzugeben.

Vokabeln ausgeben

Mit diesem Programmpunkt (Bild 2) sind Sie in der Lage, die im Speicher befindliche Datei durchzublätern, beispielsweise um sich einen Überblick über die Art der Vokabeln zu verschaffen. In den beiden Ausgabefenstern auf der lin-

Superbase Professional

Jetzt gibt es Superbase Professional! Die Profi-Version der bekannten, relationalen Datenbank Superbase mit neuen, mächtigen Features:

- Leistungsfähige Textverarbeitung mit Serienbrieffunktion
- Intelligenter Formulareditor für mehrseitige relationale Formulare mit bis zu 240 Spalten
- Mächtige, Basic-ähnliche Datenbanksprache »DML« mit Unterstützung von sämtlichen Superbase-Professional-Funktionen, Pull-down-Menüs, Eingabe-Masken, Fenstern, Scroll-Balken usw.

Superbase Professional ist das ideale Entwicklungswerkzeug - auch für komplexe Aufgaben!

Superbase Professional für Amiga

Bestell-Nr. 51672

DM 599,-*

(sFr 539,-*/öS 5990,-*)

Superbase Professional für Atari

Bestell-Nr. 51673

DM 599,-*

(sFr 539,-*/öS 5990,-*)

Upgrades:

Upgrade Superbase auf Superbase Professional für Atari

Bestell-Nr. 51673U

DM 300,-*

(sFr 280,-*/öS 3000,-*)

Upgrade Superbase auf Superbase Professional für Amiga

Bestell-Nr. 51672U

DM 300,-*

(sFr 280,-*/öS 3000,-*),

(Gegen Einsendung der

Originaldiskette und gegen Vorauskasse mit Verrechnungsscheck oder der abgedruckten Zahlkarte.)

* Unverbindliche Preisempfehlung

Fragen Sie bei Ihrem Händler nach weiteren Unterlagen.

Markt&Technik-Support:

Bei User-Registrierung rechtzeitige Update-/Upgrade-Information und Support-Unterstützung: Telefon 0 89/46 13-6 46 oder -205.

Senden Sie uns bitte Ihre Registrierungskarte.

SuperbaseTM

PROFESSIONAL



Markt&Technik-Produkte
erhalten Sie in den Fachabteilungen
der Warenhäuser, im Versandhandel,
in Computer-Fachgeschäften oder
bei Ihrem Buchhändler.

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Fragen Sie Ihren Fachhändler
nach unserem kostenlosen Gesamtverzeichnis
mit über 500 aktuellen Computerbüchern
und Software. Oder fordern Sie es direkt
beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0; Rudolf Lechner&Sohn, Heizwerkstrasse 10, A-1232 Wien, Telefon (0222) 677526; Ueberreuter Media Verlagsges.m.bH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.

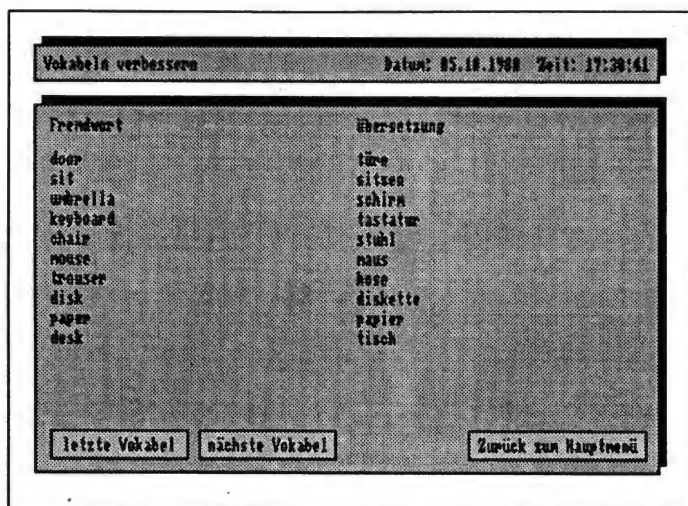


Bild 4. Hier wird die zu bearbeitende Vokabel ausgewählt

ken Seite sehen Sie das jeweilige Fremdwort und die passende Übersetzung. Das Fenster links oben bietet Informationen über die Größe der Vokabeldatei und die laufende Nummer der angezeigten Vokabel. Der Balken unterhalb der Vokabelnummer zeigt grafisch den Anteil der Vokabeldatei in Abhängigkeit von der Maximalzahl von Vokabeln an. Im letzten Fenster rechts unten können Sie durch Anwahl von »Vokabel weiter« und »Vokabel zurück« in der Datei vor- und zurückblättern. Wenn Sie mit der Maus die Zahl »10« anklicken, wird die Datei in Zehnerschritten durchgeblättert.

Statistik ausgeben

Diese Funktion zeigt, wie häufig Sie die angezeigte Vokabel in einer bestimmten Abfrageart (s. unten) gewußt haben. Während des Durchblätterns erscheint im Balken unter dem Namen »Vokabelnummer« ein dünner roter Strich, der die aktuelle Position im Vokabelfeld markiert. Mit Hilfe der Maus läßt sich dieser Strich an eine andere Stelle des Balkens setzen. Auf diese Weise können Sie sich das Durchblättern einer längeren Datei wesentlich erleichtern.

Vokabeln editieren

Der Programmteil hilft, einzelne Vokabeln zu verbessern oder zu löschen. Die im Speicher vorhandene Datei kann unter diesem Punkt auch komplett gelöscht werden. Wollen Sie einzelne Vokabeln verbessern oder löschen, klicken Sie mit der Maus die gewünschte Vokabel an (Bild 4). Bei »Vokabel verbessern« erscheint die ausgewählte Vokabel in der gleichen Eingabemaske wie bei »Vokabel eingeben«. Der Cursor wird am Ende der Voka-

bel platziert, die Verbesserungen können über Tastatur vorgenommen werden. Mit <RETURN> wird die neue Vokabel in die Datei übernommen. Durch Anwahl eines der beiden Menüpunkte »letzte Vokabel« und »nächste Vokabel« blättern Sie in der Datei.

Vokabeln suchen

Mit diesem Programmteil können Sie im Speicher oder auf Diskette eine bestimmte Vokabel finden. Soll ein Begriff auf Diskette gesucht werden, müssen Sie die zu überprüfenden Vokabeldateien auswählen. Diese Auswahl erfolgt genauso wie beim Laden oder Löschen von Dateien. Daraufhin erscheint die Aufforderung, ei-

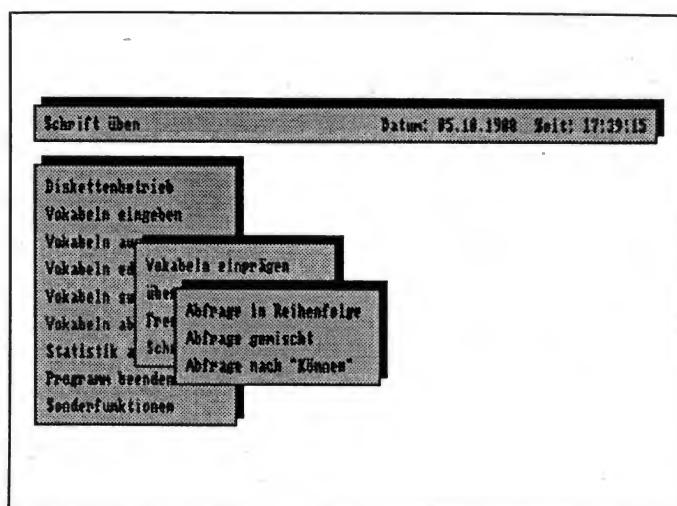


Bild 5. Diese Untermenüs erscheinen bei der Abfrage

werden ausgegeben. Im Fenster rechts oben erhalten Sie die Information, an welcher Stelle der Datei der Computer im Moment sucht. Diese Stelle wird wiederum durch einen roten Strich innerhalb der Balkenanzeige im rechten oberen Fenster markiert.

Frage und Antwort

Vokabeln abfragen

Dieser Menüpunkt (Bild 5) stellt Ihnen vier verschiedene Arten zur Verfügung, eine Vokabel zu lernen.

1. Vokabeln einprägen

Mit dieser Funktion können Sie

gleichen Programmablauf. Sie unterscheiden sich nur in der Art der Abfrage. Im linken oberen Fenster erscheint die jeweilige Vokabel. Nach einem Mausklick zeigt das Programm links unten das jeweilige Gegenstück der abgefragten Vokabel. Im rechten Fenster erhalten Sie die Information, die wievielte Vokabel aus der Gesamtdatei momentan abgefragt wird. Unten rechts befindet sich die Angabe über die Zahl der abgefragten Vokabeln. In diesem Fenster geben Sie auch an, ob die Vokabel bekannt war oder nicht. Sind Sie am Ende einer Datei angekommen, beginnt das Programm am Anfang der Datei. Brechen Sie die Abfrage mit dem Schalter »zurück zum hauptmenü« ab, werden die Statistikdaten der Vokabeldatei auf Diskette gespeichert.

4. Schrift üben

Bei dieser Funktion arbeitet das Programm ähnlich. Hier müssen Sie jedoch das Fremdwort über die Tastatur eingeben. Danach vergleicht der Computer die Eingabe mit dem gespeicherten Fremdwort. Er erkennt Eingaben als richtig, wenn diese entweder mit dem gespeicherten Wort übereinstimmen (Groß- und Kleinschreibung wird nicht unterschieden), oder wenn die Eingabe mit einem der Synonyme übereinstimmt.

Bei allen drei Arten der Abfrage können Sie vor Beginn die Reihenfolge festlegen, in der die Vokabeln abgearbeitet werden sollen. Die Vokabeln können in der eingegebenen Reihenfolge, in gemischter Reihenfolge oder nach Könnensstand abgefragt werden. Bei letzterem wählt der Computer primär die weniger gut beherrschten Vokabeln aus.

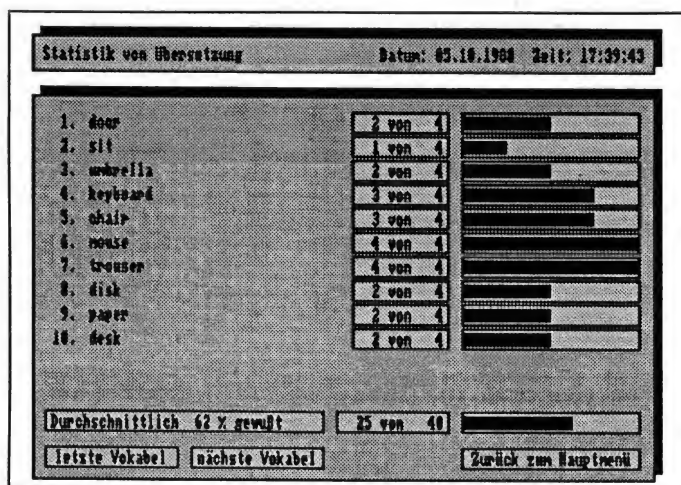


Bild 6. Statistische Auswertung einer Vokabeldatei — der Lernerfolg wird sichtbar

nen Suchbegriff einzugeben. Unter Suchbegriff ist hier eine Zeichenkette zu verstehen, die an beliebiger Stelle in der gesuchten Vokabel stehen kann. Groß- oder Kleinschreibung wird bei der Suche nicht berücksichtigt. Alle Vokabeln, die den Suchbegriff beinhalten,

sich, vor allem bei ganz neuen Vokabeln, zunächst einen Überblick verschaffen und sich die Vokabeln einprägen. Es wird noch keine statistische Speicherung vorgenommen.

2. Übersetzung abfragen

3. Fremdwort abfragen

Beide Funktionen haben den

Statistik muß sein

Der Punkt »Statistik ausgeben« (Bild 6) erstellt für jede abgefragte Vokabel eine Statistik. Diese ist in die drei Abfragearten aufgeteilt. Die Ergebnisse sind sowohl prozentual als auch grafisch dargestellt. Mit »letzte Vokabel« und »nächste Vokabel« werden die angezeigten Worte jeweils um eine Position nach oben oder unten verschoben. Weiter zeigt die Funktion den durchschnittlichen Wert aller in einer Datei vorhandenen Vokabeln an.

Sonderfunktionen

Dieser Programnteil verleiht dem Vokabeltrainer noch mehr Komfort. Die eingestellten Farbwerte können je nach Wunsch des Anwenders verändert werden. Diese Einstellung ist nur einmal nötig, da die neu eingestellten Werte mit dem entsprechenden Anwendernamen auf Diskette gespeichert werden.

1. Info zu Abfragedaten
Zeigt an, wann die geladene Datei zum letzten Mal abgefragt wurde.

2. Benutzung des Programms

Mit diesem Punkt können Sie feststellen, an welchem Tag und zu welcher Uhrzeit Sie das Programm zuletzt benutzt haben. Diese Programmfunktion verdeutlicht, ob eine Wiederholung nach längeren Pausen nötig ist.

Programm beenden

Wichtig: Falls Sie eine Übung mit dem Vokabeltrainer beenden wollen, muß in jedem Fall diese Funktion gewählt werden. Erst nach Aktivierung dieses Punktes werden die Benutzungsdaten des Anwenders auf Diskette gespeichert. Vor dem endgültigen Ende des Programms erscheint zusätzlich eine Sicherheitsabfrage. Klicken Sie daraufhin »JA« an.

Das Programm Vokabel-Trainer benötigt außer »Trainer« und »Trainer HP« noch Amiga-Basic auf der Diskette.

Sicher wird Ihnen mit Hilfe dieses Programms das Lernen von Vokabeln mehr Freude bereiten, als nach der herkömmlichen Methode.

(Andreas Regul/
Dieter Meyer/kn/rs)

Programmname:	Trainer
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic
Bemerkung:	Ladeprogramm für »Trainer HP«

```

31 1H LOCATE 20,18:PRINT "(w) 7/1988 Andreas Regul Tel.: 06
    174/3706"
32 DU g=36
33 Cs AREA (240,112)
34 uS AREA (276,76):AREA (312,112):AREA (307,112)
35 JE AREA (276,81):AREA (245,112):AREA (240,112)
36 5B AREAFILL
37 hn AREA (255,100):AREA (298,100):AREA (299,102)
38 KA AREA (256,102):AREA (256,100)
39 8E AREAFILL
40 yL AREA (274,90):AREA (279,90)
41 Yh AREA (279,126):AREA (274,126)
42 BH AREAFILL
43 Oq AREA (279,90):AREA (297,90)
44 Z5 AREA (297,92):AREA (279,92)
45 EK AREAFILL
46 dk AREA (279,110):AREA (279,108)
47 ah AREA (297,108):AREA (297,110)
48 HN AREAFILL
49 55 CIRCLE (297,100),18,3,4.71,1.57
50 1h CIRCLE (297,100),23,3,4.71,1.57
51 4z PAINT (318,100)
52 Ne LINE (298,109)-(304,109),3
53 66 AREA (281,108):AREA (317,126)
54 9w AREA (311,126):AREA (280,111)
55 OU AREAFILL
56 Eb LINE (274,99)-(279,99),2
57 07 LINE (274,103)-(279,103),2
58 CE LINE (291,90)-(293,92),2
59 Xd LINE (284,90)-(286,92),2
60 UU LINE (307,106)-(311,109),2
61 jY LINE (301,107)-(304,110),2
62 sX WHILE MOUSE(0)<0:WEND
63 la t=TIMER
64 A5 WHILE TIMER<t+6 AND MOUSE(0)=0 AND INKEY$="":WEND
65 v3 RUN "TRAINER HP"
66 xQ0 REM Farbwerte
67 TM4 DATA 0,0.04,0.1,0.16,0.22,0.29,0.35,0.41,0.47,0.54,0.6,
    0.66,0.72,0.79,0.85,0.91
68 xp DATA 10,9,7,10,9,7,13,13,13,1,1,1,16,1,1,12,10,3,1,10,1
    ,1,1,15
(C) 1988 M&T

```

Listing 1. »Trainer« ist das Ladeprogramm. Es vergrößert den Basic-Speicher für das Hauptprogramm und lädt dieses nach. Bitte geben Sie das Listing mit dem Checksummer (Seite 159) ein.

Programmname:	Trainer HP
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

Programmname: Trainer HP

```

1 wp0 REM Vokabel - Trainer Vers. 1.0
2 N9 REM von Andreas Regul (c) 7/1988
3 5W REM Initialisierung
4 rd4 DEFINT a-z
5 Du mmv=200
6 Iv DIM SHARED ta$(7),t$(100),na$(10),f!(16),farb(7,2),fa
    (10,7,2),bn$(10,2),abn$(10,2)
7 Is DIM SHARED d$(101),da$(101),dm(101),ff(10),v$(mmv+1,1
    )
8 yu DIM SHARED st(mmv+1,5),re(mmv+1),re1(mmv+1),vo(mmv+1)
9 Yw FOR i=1 TO 7
10 iG6 READ ta$(i)
11 GL4 NEXT
12 dR FOR i=1 TO 100

```

Listing 2. Das Hauptprogramm »Trainer HP« muß sich auf der Diskette im Laufwerk befinden, das als aktueller Pfad angegeben ist. Ansonsten erscheint »File not found«. Bitte mit dem Checksummer (Seite 159) eingeben.

Fortsetzung des Listings auf Seite 148

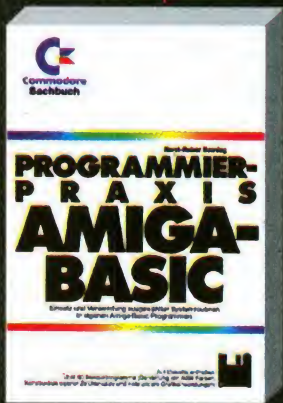
Programmname: Trainer

```

1 EJO REM Ladeprogramm fuer Vokabel-Trainer
2 61 REM von Andreas Regul (c) 7/1988
3 5W REM Initialisierung
4 y14 CLEAR,100000&
5 FA DIM f(16),fa(7,2)
6 dH FOR i=1 TO 16
7 dj6 READ f(i)
8 DI4 NEXT
9 000 REM Bildschirm Aufbau
10 iX4 SCREEN 1,640,244,3,2
11 vz WINDOW 1,"Vokabel - Trainer Vers. 1.0
    von Andreas Regul (c) 7/1988",(0,0)-(631,230),0,1
12 Sn FOR i=1 TO 4
13 nl6 MENU i,0,0,""
14 JO4 NEXT
15 by FOR i=0 TO 7
16 Th6 FOR j=0 TO 2
17 V48 READ fa(i,j)
18 NS6 NEXT
19 2a PALETTE i,f(fa(i,0)),f(fa(i,1)),f(fa(i,2))
20 PU4 NEXT
21 Ry LINE (100,40)-(520,166),2,bf
22 2N LINE (100,40)-(520,166),3,b
23 OP LINE (101,40)-(101,166),3
24 OR LINE (519,40)-(519,166),3
25 72 LINE (108,36)-(527,39),3,bf
26 Hu LINE (521,40)-(527,162),3,bf
27 So COLOR 3,2
28 nP LOCATE 7,18:PRINT "V O K A B E L - T R A I N E R Vers
    ion 1.0"
29 TQ LOCATE 10,26:PRINT "v o n"
30 Br LOCATE 16,44:PRINT "S o f t w a r e"

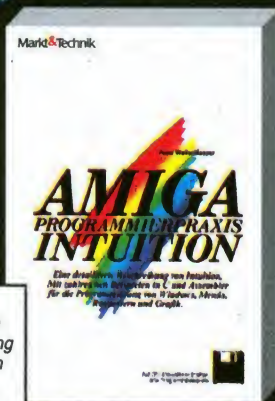
```


Brandneue Bücher für Ihre **Amiga-** **Bibliothek**



H.R. Henning
Programmierpraxis Amiga-Basic
Einsatz und Verwendung ausgewählter Systemroutinen in eigenen Amiga-Basic-Programmen. Die beigefügte Diskette enthält über 80 Beispielprogramme.
1988, 368 Seiten, inkl. Diskette
Bestell-Nr. 90549
ISBN 3-89090-549-8
DM 59,-/sFr 54,30/£S 460,20

P. Wollschlaeger, Amiga:
Programmierpraxis Intuition
Eine detaillierte Beschreibung von Intuition. Mit zahlreichen Beispielen auf Diskette.
1988, 330 Seiten, inkl. Disk.
Bestell-Nr. 90593
ISBN 3-89090-593-5
DM 69,-/sFr 63,50/£S 538,20



A. Plenge
Amiga-3-D-Grafik und Animation
Eine leichtverständliche Anleitung für die Erstellung von dreidimensionalen Grafiken: Clipping, Perspektivische Projektion, Raytracing, Versteckte Linien, Schatten, Reflexion, 3-D-Editor.
1988, 376 Seiten, inkl. Diskette
Bestell-Nr. 90526, ISBN 3-89090-526-9
DM 69,-/sFr 63,50/£S 538,20

D. Myers, Amiga: Grafik * Musik * DFÜ
Leichtverständlicher Programmierkurs für die erfolgreiche Grafik- und Soundprogrammierung in Amiga-Basic. Die Musikfunktionen, die Stimme des Amiga, Grafik, Animation, Datenfernübertragung und viele weitere Themen werden detailliert beschrieben.
1988, 231 Seiten, inkl. Diskette
Bestell-Nr. 90579, ISBN 3-89090-579-X
DM 59,-/sFr 54,30/£S 460,20



I. Krüger, Amiga: Programmieren mit Modula 2
Leichtverständlicher Modula-2-Kurs. Mit vielen Beispielen für die systemnahe Programmierung unter der grafischen Benutzeroberfläche »Intuition«. Auf der Diskette enthalten: alle Modula-2-Beispiele für Screens, Windows etc.
1988, 362 Seiten, inkl. Diskette
Bestell-Nr. 90554, ISBN 3-89090-554-4
DM 69,-/sFr 63,50/£S 538,20



P. Wollschlaeger, Amiga-Assembler-Buch
Dieses Buch beweist, daß Assembler-Programmierung ganz einfach ist: Ein 68000er-Kurs mit vielen Beispielen. Mit ausführlichem Verzeichnis aller Systemroutinen, Anleitung für das Einbinden von Assembler-Routinen in Amiga-Basic und vielen Informationen über die Internas des Amiga-Betriebssystems. Mit Beispieldiskette.
1987, 329 Seiten, inkl. Diskette
Bestell-Nr. 90525, ISBN 3-89090-525-0
DM 59,-/sFr 54,30/£S 460,20



H.R. Henning, Programmieren mit Amiga-Basic
Eine gründliche Einführung in die Programmierung mit Amiga-Basic: Animation (bewegte Grafiken und Sprites) - Befehle zur Sprach- und Musikausgabe - Fenstertechnik - Sequentielle Dateiverwaltung - Spieleprogrammierung - viele Tips&Tricks und eine 3 1/2"-Diskette mit über 100 Programmbeispielen.
1987, 363 Seiten, inkl. Diskette
Bestell-Nr. 90434, ISBN 3-89090-434-3
DM 59,-/sFr 54,30/£S 460,20

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computerfachgeschäften oder in den Fachabteilungen der Warenhäuser.

Irrtümer und Änderungen vorbehalten.


Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2,
8013 Haar bei München, Telefon (089) 4613-0.

SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656.

ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0.

Rudolf Lechner & Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526

Überreuter Media Verlagsges.m.bH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.



Fragen Sie bei Ihrem Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

ANWENDUNGEN

Die ersten Schritte

Nach dem Start mit »RUN« werden Sie zuerst nach dem Namen der Kursdatei gefragt (siehe oben). Wollen Sie den vorgegebenen Namen verwenden, drücken Sie nur <RETURN>. Anschließend wird die Kursdatei geladen. Wenn sie unter diesem Namen noch nicht existiert, werden Sie gefragt, ob eine neue angelegt werden soll. Beantworten Sie mit »n«, können Sie den Namen nochmals eingeben. Tippen Sie »j« (was Sie beim ersten Start tun sollten), legt das Programm eine neue Datei an. Falls Sie eine bereits existierende Kursdatei überschreiben wollen, müssen vor dem eigentlichen Filenamen drei Sterne und ein Leerzeichen eingegeben werden. Achtung: Alle Daten in dieser Datei gehen dabei verloren! Wichtig ist, daß Sie die Diskette mit der Kursdatei immer im Laufwerk lassen, solange Sie mit dem Programm arbeiten.

Sie befinden sich nun im Hauptmenü (Bild 1). Dabei wird Ihnen auffallen, daß das ganze Programm über die Tastatur gesteuert wird. Schließlich soll deren Bedienung ja erlernt werden. Zudem kann die Bedienung auch ohne Maus komfortabel und einfach sein.

Einige Menüpunkte, die im Moment noch nicht sinnvoll sind, fehlen und sind nicht wählbar. Im folgenden werden alle Menüpunkte der Reihe nach besprochen. Beginnen Sie jedoch nicht sofort, den Kurs durcharbeiten. Das sollten Sie erst dann tun, wenn Sie auch die Hinweise zur Durchführung des Kurses gelesen haben. Dies soll Sie aber nicht daran hindern, die Funktion des Programms beim Lesen der Bedienungsanleitung auszuprobieren.

<F1> - Lektion/Textübung wählen

Mit der Funktionstaste <F1> gelangen Sie vom Hauptmenü in ein Untermenü, in dem alle 20 Lektionen aufgelistet sind. Wählen Sie eine davon aus, indem Sie die jeweilige Zahl, gefolgt von <RETURN>, eingeben. Im Anschluß daran werden alle Lektionen in den Speicher geladen. Wenn Sie später weitere Lektionen anwählen, erfolgen keine weiteren Zugriffe auf die Diskette. Wenn Sie keine reguläre Lektion bearbeiten wollen, sondern einen beliebigen ASCII-Text üben wollen,

drücken Sie statt einer Zahl nur die RETURN-Taste und Sie werden nach dem Filenamen des Textes gefragt. Tippen Sie den Namen des gewünschten Files ein. Nach dem Ladevorgang gelangen Sie wieder ins Haupt-Menü.

<F2> - Lektion starten

Je nachdem, ob Sie eine Lektion oder einen Übungstext geladen haben, hat dieser Menüpunkt den Titel »Lektion xx starten« oder »Textübung starten«. Nach Betätigen von <F2> werden zunächst einige Daten aus der Kursdatei geladen und der wichtigste Teil des Programms beginnt.

Auf dem Bildschirm wird ein Abbild einer Amiga-Tastatur, die an jene des Amiga 500 angelehnt ist, gezeichnet (Bild 2). Die Querlinien mitten im Tastenfeld sollen am Anfang helfen, alle Tasten mit dem richtigen Finger anzuschlagen. Die Hilfslinien trennen also den »Zuständigkeitsbereich« der einzelnen Finger voneinander. Die helle Linie in der Mitte stellt die Trennung zwischen rechter und linker Hand dar. Wenn in den Voreinstellungen (<F7>) das Zeichnen der Hilfslinien nicht angeschaltet ist, sind diese natürlich nicht zu sehen. Genauso verhält es sich mit der Hilfstastatur und der Anzeige der jeweils zu drückenden Taste (siehe unten).

Der Bildschirmaufbau benötigt einige Sekunden, jedoch nur beim ersten Aufruf dieses Programmtells. Danach ist die Grafik im RAM gespeichert. Ganz oben sehen Sie die Statuszeile, in der Sie lesen können, welche Lektion Sie gerade bearbeiten, welche Nummer die aktuelle Zeile hat und in welchem Teil der Lektion (Griff-, Wort- oder Satzübung) Sie sich im Moment befinden. Darunter wird die Anzahl der Fehler angezeigt, die Sie in der aktuellen Zeile gemacht haben. Noch weiter unten sehen Sie in einem farbigen Feld den Klartext der jeweiligen Zeile. Unter dieser befindet sich ein nach oben zeigender Pfeil, der auf das Zeichen zeigt, das eingetippt werden muß.

Im unteren Teil des Bildschirms befindet sich noch ein gelber Kasten, der als Hilfe für den Benutzer die möglichen Tasten am Zeilenende beinhaltet. Ist der Bildschirmaufbau abgeschlossen, ertönt ein Summen, das zeigt, daß es losgehen kann. Sie müssen nun Buchstaben für Buchstaben aus der angezeigten Textzeile abtippen. Der Pfeil wandert dabei von Zeichen zu Zeichen

mit. Um Sie davon abzuhalten, spärende Blicke auf die Computertastatur zu werfen, werden die zu drückenden Tasten auf der Hilfstastatur hervorgehoben (sofern in der Voreinstellung diese Funktion aktiviert wurde).

Machen Sie einen Fehler, hören Sie einen Piepston und der Pfeil verwandelt sich unter dem falsch getippten Zeichen in ein Ausrufungszeichen (!«). Die Fehleranzeige wird um eins erhöht. Tippen Sie nun das Zeichen richtig, bekommt der Cursor (Pfeil) wieder sein normales Aussehen und Sie können weitertippen. Am Zeilenende angelangt erscheint an Stelle des Cursors ein Linkspfeil »-« und Sie können nun die RETURN-Taste drücken, um in die nächste Zeile zu gelangen, die ESC-Taste, um die Lektion ohne Speichern der Ergebnisse abzubrechen oder <+>, um die getippte Zeile nochmals einzugeben. In der Gesamtwertung werden dann nur jene Fehler, die beim weiteren Durcharbeiten der Zeile gemacht wurden, berücksichtigt.

Um die Funktionsweise auszutesten und sich daran zu gewöhnen, sollten Sie einmal eine Zeile abtippen, am Zeilenende aber nicht die RETURN-Taste drücken, sondern <ESC>. Schließlich handelte es sich nur um einen Versuch, und ein Speichern der Ergebnisse ist wohl kaum erwünscht.

Der Computer weiß alles...

Am Ende jeder Lektion werden die erzielten Ergebnisse, sowie allgemeine Angaben zur Lektion, am Bildschirm angezeigt (Bild 3) und in die Kursdatei gespeichert. So weiß das Programm immer, welche Lektion Sie schon durchgearbeitet haben, wieviel Fehler Sie dabei gemacht haben und wie schnell Sie beim Tippen waren. Genauso ist der Computer — und damit auch Sie — ständig im Bilde darüber, wieviel prozentual falsch gemacht wurde und wie viele Anschläge pro Minute Sie erreicht haben. Schließlich ist er auch noch darüber informiert, ob und wie oft Sie einzelne Lektionen wiederholt und sich dabei verbessert oder verschlechtert haben. Erwähnenswert ist hierbei auch die grafische Anzeige des Fehlerprozentsatzes. All das schauen Sie sich am besten erst an, wenn Sie mit dem eigentlichen Kurs ernsthaft be-

Video-Scape 3D - Berechnete Realität

Mit Video-Scape 3D können Sie dreidimensionale Objekte aus verschiedenen Blickwinkeln betrachten und durch Hinzufügen von Kamerafahrten und frei wählbarem Lichteinfall einen realistischen Computer-Videofilm erstellen.

Bestell-Nr. 51671

DM 385,-* (sFr 345,-*/öS 3850,-*)

Aegis Images -

Farbenpracht leichtgemacht

Ein ideales Standard-Zeichenprogramm mit über 4000 Farben.

Bestell-Nr. 54108

DM 69,-* (sFr 62,-*/öS 690,-*)

Aegis Animator und Images - Bringt Bewegung in Ihre Bilder

Aegis Animator verbindet drei Animationstechniken, damit Sie vielseitige Desktop-Video-Produktionen auf dem Amiga erzeugen können!

Bestell-Nr. 54109

DM 249,-* (sFr 225,-*/öS 2490,-*)

Aegis Draw

Ein leicht zu bedienendes Zeichenprogramm.

Bestell-Nr. 54106

DM 199,-* (sFr 179,-*/öS 1990,-*)

Aegis Draw Plus - CAD: Ein Traum wird erschwinglich

Computerunterstütztes Konstruieren von strukturierten Grafiken.

Bestell-Nr. 54107

DM 385,-* (sFr 345,-*/öS 3850,-*)

Aegis Video-Titler -

Verwandeln Sie Ihren Amiga in eine leistungsfähige Titelmaschine

Ein Text- und Grafikgenerator.

Bestell-Nr. 54101

DM 249,-* (sFr 225,-*/öS 2490,-*)

Aegis Impact - Verleihen Sie Ihrer Präsentation einen Ausdruck, der Eindruck macht

Das Desktop-Programm zur Erstellung von Präsentationsgrafiken.

Bestell-Nr. 54104

DM 149,-* (sFr 135,-*/öS 1490,-*)

Aegis Sonix - Wetten, daß auch Sie mit diesem Programm einen Hit schreiben?

Synthesizer- und Kompositionsprogramm.

Bestell-Nr. 54105

DM 149,-* (sFr 135,-*/öS 1490,-*)

Aegis AudioMaster - Das Tonstudio für den Amiga

Nachbearbeitung von digitalisierten Klängen.

Bestell-Nr. 54103

DM 99,-* (sFr 89,-*/öS 990,-*)

Aegis Diga -

Der Schlüssel zur modernen Datenkommunikation

Telekommunikationssoftware.

Bestell-Nr. 54102

DM 149,-* (sFr 135,-*/öS 1490,-*)

* Unverbindliche Preisempfehlung

Markt&Technik-Support:

Bei User-Registrierung rechtzeitige Update-/Upgrade-Information und Support-Unterstützung. Senden Sie uns bitte Ihre Registrierungskarte.



AEGIS: Kreative Software für den Amiga.

Markt&Technik-Produkte
erhalten Sie in den Fachabteilungen
der Warenhäuser, im Versandhandel,
in Computer-Fachgeschäften oder
bei Ihrem Buchhändler.

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Fragen Sie Ihren Fachhändler
nach unserem kostenlosen Gesamtverzeichnis
mit über 500 aktuellen Computerbüchern
und Software. Oder fordern Sie es direkt
beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 587 1393-0; Rudolf Lechner & Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 67 75 26; Ueberreuter Media Verlagsges.m.b.H (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 48 15 43-0.

ANWENDUNGEN

Aniga Keyboard Master V2.3 by Robert Kretzschmar © 1988 Markt & Technik	
ENDE von Lektion 1	
Ergebnis	
Anschläge:	662
Fehler:	49
Fehlerquote:	7.40 %
benötigte Zeit:	9 Minuten, 35 Sekunden
Anschläge pro Minute:	69
Wiederholung der Lektion:	keine
Drücken Sie eine Taste	

Bild 3. Die Auswertung gibt Aufschluß über den Lernerfolg und Ihren momentanen Leistungsstand

ginnen wollen, denn die Bedienung dieses Programmteils erklärt sich von selbst.

<F3> - Übungstext eingeben

Dieses Lernprogramm gibt Ihnen nicht nur die Möglichkeit, fertige Lektionen zu bearbeiten, es können vielmehr auch eigene Texte eingegeben werden, die Sie zur Übung abtippen können. Dieser Programmteil ist ebenfalls weitgehend selbsterklärend. Es bleibt nur zu bemerken, daß Sie die Eingabe der Zeilen (maximal 20 Zeilen, je Zeile höchstens 78 Zeichen) mit einer Leerzeile abschließen müssen, um ins Hauptmenü zu gelangen. Dort werden Sie bemerken, daß unter »F2« nun »Textübung starten« angeführt ist. Schließlich wollen Sie jetzt Ihren Text bearbeiten, nicht die Lektion. Das selbe gilt beim Laden eines Übungstextes (siehe oben).

<F4> - Lektions-/Übungstext anzeigen

Über diesen Menüpunkt wird der aktuelle Lektionstext beziehungsweise der eingegebene oder geladene Übungstext angezeigt. Dabei fallen hier die violetten Zahlen am Zeilenanfang auf. Diese zeigen an, wie oft jede Zeile abgetippt werden muß.

Bei der Eingabe eines Übungstextes wird übrigens automatisch eine »1« vor jede Zeile geschrieben. Auch wenn Sie eine ASCII-Datei laden (Menüpunkt 1), wird jede Zeile, die nicht mit einer Ziffer beginnt, mit einer »1« am Zeilenanfang ergänzt. Bei den Lektionstexten können Sie mit einem ASCII-Editor die Anzahl der Wiederholungen leicht ändern. Beachten Sie jedoch,

daß nur die erste Ziffer als Wiederholungszahl erkannt wird. Diesen Programmteil verlassen Sie, indem Sie eine beliebige Taste betätigen.

<F5> - Übungstext speichern

Dieser Menüpunkt kann nur dann gewählt werden, wenn Sie zuvor einen Übungstext geladen oder eingegeben haben. Das Programm fragt Sie nach dem gewünschten Filenamen. Wollen Sie den Text doch nicht speichern, drücken Sie nur <RETURN>.

<F6> - Auswertung

In diesem Programmteil werden die Daten, die die Kursdatei über die einzelnen Lektionen enthält, angezeigt (Bild 4). Lektionen, die noch nicht bearbeitet worden sind, werden dabei dunkel dargestellt. Die angewählte Lektion wird besonders gekennzeichnet. Das Feld »Differenz FQ« enthält, wenn die Lektion mehr als einmal bearbeitet worden ist, die Differenz des neueren Fehlerquotienten (FQ) zum älteren. Hier sehen Sie, ob Sie sich beim Wiederholen verbessert oder verschlechtert haben. Direkt daneben befindet sich eine weitere Spalte, die Ihnen noch unbekannt ist: Unter »Punkte« wird für jede Lektion eine bestimmte Zahl angegeben, die etwa mit einer Note vergleichbar ist. Bei Punktwerten unter 50 werden dahinter drei Ausrufungszeichen geschrieben, was bedeutet, daß Sie diese Lektion unbedingt wiederholen sollten. Werte über 100 sind sehr gut. In die Punktwertung geht neben einigen anderen Faktoren hauptsächlich der Fehlerquotient ein.

Aniga Keyboard Master V2.3 by Robert Kretzschmar @ 1988 Markt & Technik

Auswertung (Kursdatei df0:test)

Lektion	Wiederh.	Fehlerquotient	Anschl./min	Differenz FQ	Punkte
1	0 mal	7.40 %	69		90
2	0 mal	5.47 %	177		112
3					
4					
5					
6	0 mal	3.81 %	150		111
7					
8					
9					
10					
11					
12					
13					
14	0 mal	6.58 %	118		97
15					
16					
17					
18					
19					
20					

Taste drücken

Bild 4. Durch das Anlegen einer Kursdatei auf Diskette wissen Sie ständig über Ihren »Wissens-Status« bescheid

<F7> - Voreinstellungen

Mit <F7> gelangen Sie in den Programmteil, in dem Sie verschiedene Voreinstellungen (vergleichbar mit Preferences der Workbench) ändern können. Diese Änderungen werden in die Kursdatei geschrieben und bei jedem erneuten Laden der Kursdatei im Programm berücksichtigt.

Sie haben sechs Möglichkeiten, Einstellungen und damit den Programmablauf zu verändern. Es handelt sich dabei um »Schalter«, wobei durch Druck auf die links neben der jeweiligen Bezeichnung stehende Zahlentaste zwischen zwei Stellungen hin- und hergeschaltet werden kann. Bei einigen Schalterkombinationen nimmt das Programm selbstständig Änderungen der Stellungen vor, das heißt es verhindert unlogische Kombinationen. Kommen wir nun zur Bedeutung der einzelnen Schalter:

Wenn Schalter 1 ausgeschaltet ist, wird die Hilfstastatur nicht mehr angezeigt.

Schalter 2 ist für die Hervorhebung der jeweils zu drückenden Tasten auf der gezeichneten Hilfstastatur zuständig.

Mit Schalter 3 wird bestimmt, ob die Textzeile immer angezeigt wird oder nicht. Es ist sinnvoll, diese auszuschalten, wenn man sich voll auf die Griffe und die Hilfstastatur konzentrieren möchte.

Ist Schalter 4 ausgeschaltet, wird das Zeichnen der Hilfslinien auf der Tastatur unterdrückt, da diese mit der Zeit störend wirken können.

Wenn Sie das automatische Wiederholen einzelner Zeilen unterdrücken wollen, müssen

Sie Schalter 5 ausschalten. Dies ist dann sinnvoll, wenn Sie eine Lektion schon oft bearbeitet haben und diese nur kurz auffrischen wollen.

Mit dem sechsten und letzten Schalter wählen Sie schließlich das Default-Laufwerk (df0: oder df1:), das immer dann verwendet wird, wenn bei Filenamen keine Laufwerksangabe gesondert angegeben wird. Auch die Lektionstexte werden stets von diesem Laufwerk geladen. Bitte verwechseln Sie diese Einstellung nicht mit dem Default-Laufwerk, das Sie vor dem ersten Programmstart im Listing direkt eingestellt haben. Dieses Laufwerk gilt nur für die Kursdatei, ansonsten hat das in diesem Programmteil einzustellende Laufwerk Priorität.

Mit <ESC> werden eventuelle Änderungen in die Kursdatei gespeichert und Sie kommen zurück ins Hauptmenü.

ESC - Programm beenden

Mit <ESC> im Haupt-Menü verlassen Sie das Programm.

Der Kurs beginnt

Ich hoffe, Sie haben nun die ersten Gehversuche mit Keyboard-Master hinter sich und brennen darauf, endlich den Kurs zu beginnen.

Dazu greifen Sie zunächst, ohne das Programm zu starten, die Grundstellung, die Sie in Tabelle 1 (Seite 35) finden. Diese Fingerstellung ist sehr wichtig, prägen Sie sich diese ein und versuchen Sie mehrmals, diese zu finden, ohne auf die Tastatur zu sehen.

Sie sollten nun das Programm erneut starten und einen beliebigen Filenamen für

PROGRAMM-SERVICE

Direkt bestellen statt abtippen!

Die aktuelle Diskette zum Heft:

Amiga Sonderheft 2: Grafik, Anwendung

Object-Editor: Animierte Figuren, beispielsweise für eigene Spiele, entwickeln Sie mit diesem Editor auf komfortable Weise. Sogar mit Deluxe Paint erstellte Pinsel lassen sich einlesen.

Haushaltsbuch: Mit diesem hervorragenden Anwendungsprogramm verwalten Sie alle Einnahmen und Ausgaben auf übersichtliche Weise. Eine Monats- oder Jahresstatistik zeigt, in welchen Bereichen Sie zukünftig sparen können. Jetzt haben Sie Ihre Finanzen im Griff.

Keyboard-Master: Lernen Sie im Zehn-Finger-System zu tippen. Mit diesem didaktisch ausgereiften Programm ist dies kein Problem. Für Programmierer sind sogar Spezial-Lektionen mit wichtigen Sonderzeichen vorhanden.

FastLoadCopy: Dieses Tool bringt den DIR-Befehl auf Trab. Nach der »Operation« wird das Inhaltsverzeichnis einer Diskette im D-Zug-Tempo eingelesen. Zusätzlich kopiert das Programm Disketten und versieht diese mit dem schnellen Directory.

Weiterhin befinden sich auf der Diskette alle Programme, die im Inhaltsverzeichnis des Amiga-Sonderhefts 2 mit einem Diskettensymbol gekennzeichnet sind.

3 1/2"-Diskette für Amiga

Bestell-Nr. 45802

DM 29,90 * (sFr 24,90 */öS 299,-*)
* Unverbindliche Preisempfehlung

Amiga-Hardware-Service:

Die Wiederbelebung für die C64-Peripherie

Viele Amiga-Besitzer haben noch einen C64 mit Peripheriegeräten zu Hause stehen. Mit ein bißchen Hard- und Software können Sie diese zu neuem Leben erwecken und Ihre Daten so weiterbenutzen. Dabei ist die Bedienung wirklich einfach.

Der fertig aufgebaute IEC-Handler erlaubt es, alle C64-Geräte wie die Floppy 1541 oder 1571, Commodore-MPS-Drucker und natürlich auch den C64 (zur Datenübertragung) am Amiga zu betreiben.

Das Gesamtpaket besteht aus der fertig aufgebauten Platine mit Verbindungskabel, der Treibersoftware auf 3 1/2-Zoll-Diskette sowie einer entsprechenden Dokumentation.

Bestell-Nr. 39101

DM 79,- * (sFr 71,- */öS 790,-*)
* Unverbindliche Preisempfehlung

Der »IEC-Handler« ist ab November 1988 lieferbar.

Bestellungen bitte nur gegen Vorkasse bei:

Markt&Technik Verlag AG

– Buchverlag –

IEC-Handler

Hans-Pinsel-Straße 2, 8013 Haar bei München



Weitere Angebote
auf der Rückseite!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0; Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526, Ueberreuter Media Verlagsges. mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0

AMIGA PROGRAMMSERVICE

Sie suchen hilfreiche Utilities und professionelle Anwendungen für Ihren Computer? Sie wünschen sich gute Software zu vernünftigen Preisen? Hier finden Sie beides! Unser stetig wachsendes Sortiment enthält interessante Listing-Software für alle gängigen Computertypen. Jeden Monat erweitert sich unser aktuelles Angebot um eine weitere interessante Programmsammlung für jeweils einen Computertyp. Bei Fragen zu Bestellung und Versand der Programmservice-Disketten wählen Sie bitte: Telefon (089) 46 13-232.

Bestellungen bitte nur gegen Vorauskasse an: Markt & Technik Verlag AG, Unternehmensbereich Buchverlag, Hans-Pinsel-Straße 2, D-8013 Haar, Telefon (089) 46 13-0. Schweiz: Markt & Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 41 56 56. Österreich: Markt & Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 587 13 93-0. Microcomputing, E. Schiller, Fasongasse 24, A-1030 Wien, Telefon (0222) 78 56 61; Bücherzentrum Meidling, Schönbrunner Straße 261, A-1120 Wien, Telefon (0222) 83 31 96. Ueberreuter Media, Verlagsges. mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 48 15 43-0. Bestellungen aus anderen Ländern bitte nur schriftlich an: Markt & Technik Verlag AG, Abt. Buchvertrieb, Hans-Pinsel-Straße 2, D-8013 Haar. Nur gegen Bezahlung der Rechnung im voraus.

Bitte verwenden Sie für Ihre Bestellung und Überweisung die beigeheftete Postgiro-Zahlkarte, oder senden Sie uns einen Verrechnungsscheck mit Ihrer Bestellung. Sie erleichtern uns die Auftragsabwicklung, und dafür berechnen wir Ihnen keine Versandkosten.

Amiga Sonderheft 1: Bauen Sie Ihr eigenes Tonstudio

Digisoft Plus: Die Software zu unserem Selbstbau-Digitizer, die den Amiga zum digitalen Sound-Studio macht. Das Amiga-Tonstudio arbeitet mit vier getrennten Tonspuren und besitzt neben der Digitalisierung von Klängen viele weitere Features. So können einzelne Spuren oder Teile daraus beliebig gemischt, geschnitten oder verbunden werden. Die grafische Ausgabe der digitalisierten Sounds macht die Bearbeitung zum Kinderspiel.

Suremosch: Ein Strategiespiel der neuen Dimension: Begeben Sie sich in eine ferne, fremde Welt und befreien Sie die Bevölkerung von bösen Wesen, die den Zeitfluß verlangsamen.

Dcopy: ist ein schnelles Kopierprogramm der Extraklasse. Neben hoher Geschwindigkeit bietet es die Möglichkeit, bis zu drei Kopien in einem Arbeitsgang anzufertigen.

Biorhythmus: Neben einer sehr schönen grafischen Darstellung der Kurven von Körper, Geist und Seele erlaubt das komfortable Programm auch die Ausgabe einer Jahresstatistik. Sie erfahren alles über Ihre »guten« und »kritischen« Tage.

3 1/2"-Diskette für Amiga

Bestell-Nr. 45801 **DM 29,90*** sFr 24,90*/öS 299,-*

Amiga 5/88: Vom Spiel zum nützlichen Utility

Diesmal finden Sie auf unserer Programmservice-Diskette wieder ein breites Spektrum an Listings. Von Spielen über Werkzeuge bis zu Anwendungen ist alles vorhanden.

Kniffel: Ein grafisch gut aufgemachtes Spiel für bis zu vier Teilnehmer. Kniffel wird sicher nicht langweilig. Ein Muß für alle Glücksspieler.

Manager: Verschafft Ihnen die Übersicht über Ihre Ausgaben in klarer Form und hilft somit Geld sparen. Komfortable Bedienung per Maus ist selbstverständlich.

CrossRef: Hilft Ihnen beim Analysieren von Programmen. Viele wichtige Daten von Basic-Programmen wie Labels und Variablen erhalten Sie schwarz auf weiß ausgedruckt. Ein unentbehrliches Hilfsmittel für Basic-Programmierer.

3-D-Tic-Tac-Toe: Ein gutes Auge und einen scharfen Verstand brauchen Sie für diese dreidimensionale Spielvariante.

Recover: Rettet versehentlich gelöschte Dateien von Ihrer Diskette. Auch teilweise zerstörte Dateien werden soweit als möglich restauriert.

3 1/2"-Diskette für Amiga

Bestell-Nr. 48805 **DM 29,90*** sFr 24,90*/öS 299,-*

Amiga 4/88 3-D-Landschaften aus dem Computer

Fraktalberge: Ein Muß für alle Fans von zufallserzeugten Grafiken. Fantastisch einfach in der Bedienung und sehr schnell.

Transfer: Überträgt Bilder vom C64 auf den Amiga. Mit guter Software und leicht nachzubauender Hardware.

DiskSpy: Direktes Ändern von Daten auf der Diskette ist mit diesem Werkzeug kein Problem mehr. Es stehen viele Befehle zur Verfügung.

ColorChange: Ein Basic-Unterprogramm, mit dem Sie einfach und schnell Ihre Wunschfarben auf beliebigen Bildschirmen einstellen können.

Troof: Ein spannendes Spiel in Basic mit starker Grafik und vielen verschiedenen Levels.

3 1/2"-Diskette für Amiga

Bestell-Nr. 48804 **DM 29,90*** sFr 24,90*/öS 299,-*

Amiga 3/88 Bildschirmfüllende Boot-Bilder mit allen Extras

BootGirl: Fantastische Bilder sofort nach dem Reset. Bis zu 32 Farben mit Color-Cycling. Die Bilder können auch bildschirmfüllend ohne Rand sein. Ein absolutes Muß für jeden Amiga-Besitzer.

CassCover: Selbstgedruckte Kassettenhüllen geben Ihnen den richtigen Überblick. Einfache Bedienung macht das Eingeben und Ausdrucken zur wahren Freude.

Command: Das Programm ermöglicht die Steuerung des Aztec-C-Compilers mit der Maus. Keine langen Eingaben per Tastatur, sondern ein einziger Mausclick startet nun die Übersetzung.

VideoText: Ein unentbehrliches Werkzeug für alle Video-Fans, die ihren eigenen Vorspann mit dem Amiga generieren wollen. Laufbänder, verschiedene Schriften und IFF-Bilder sind nur einige Stichpunkte, die das Programm so interessant machen.

3 1/2"-Diskette für Amiga

Bestell-Nr. 48803 **DM 29,90*** sFr 24,90*/öS 299,-*

Amiga 12/87 Super-Kopierprogramm mit viel Komfort

DCopy: Unser Programm des Monats, ein Kopierprogramm, das alles bietet, was man sich nur wünschen kann. Einige Fähigkeiten: Bis zu vier Laufwerke werden verwendet, Mehrfachkopien, abschaltbares Verify und vieles mehr.

SpeedHc: Eine sehr schnelle Hardcopyroutine für Schwarzweißausdrucke mit höchster Qualität. Leicht an andere Drucker anzupassen.

Sternenhimmel: Ein unentbehrliches Werkzeug für alle Himmelsbeobachter. Das Programm zeigt alle Sterne und Planeten von jedem beliebigen Punkt der nördlichen Hemisphäre.

Checkie42: Der Checksummer für alle Programmiersprachen von Assembler über Basic bis zu C. Ab dieser Ausgabe finden Sie bei jedem Listing die Prüfziffern.

Joy: Ein sehr kurzes und schnelles C-Programm zur Abfrage des Joysticks. Es ist leicht in eigene Programme einzubinden.

Amiga-Shell: Ein C-Programm, das Komfort ins CLI bringt. Editieren der Befehlszeile, Funktionstastenbelegung und Aliasnamen sind nur einige Fähigkeiten dieses fantastischen Programms.

3 1/2"-Diskette für Amiga

Bestell-Nr. 48705 **DM 29,90*** sFr 24,90*/öS 299,-*

Amiga 10/87 Super-Malprogramme für alle Amiga-Computer

Rainbow-Drawer: Dieses Programm des Monats bietet leistungsfähige Befehle und Funktionen, wie sie von professionellen Programmen bekannt sind: bis zu 32 Farben, alle Auflösungen, viele Befehle zum Zeichnen sowie FILL mit Mustern, BOW und anderem.

Turtle: Mit dieser Befehlserweiterung verfügen Sie über die Grafikbefehle, die bei Logo bekannt und beliebt sind.

Fractals: Dreidimensionale, realistische Gebirge mit Schattierung erzeugt dieses Programm.

Clouds: Genauso wirklichkeitsnah wie die Gebirge, aber noch erstaunlicher, sind die Wolken, die Sie mit Clouds generieren.

Apfelmännchen: Hiermit erzeugen Sie schöne Grafiken aus der beliebten Mandelbrot-Ebene.

Kudiplo: Ein gutes, unverzichtbares Werkzeug für die Kurvendiskussion stellt »Kudiplo« dar.

Senso: Testen Sie mit dieser Computer-Adaption des bekannten Spiels Ihr Gedächtnis!

Division: Bis zu 32000 Nachkommastellen können durch dieses Programm berechnet werden.

Alert: Alarme, zum Beispiel die bekannten Guru-Meditations, können Sie nun selbst programmieren. Das Programm ist in erster Linie für C-Programmierer aufschlußreich.

Border: lassen Sie den Fensterrahmen des CLI-Fensters einfach verschwinden!

SCD: Mit diesem Utility können Sie den Pfadnamen in der Titelleiste des Fensters anzeigen.

3 1/2"-Diskette für Amiga

Bestell-Nr. 48703 **DM 29,90*** sFr 24,90*/öS 299,-*

* Unverbindliche Preisempfehlung

Übrigens: Mit den Gutscheinen aus dem »Super-Software-Scheckheft« für DM 149,- können Sie sechs Software-Disketten Ihrer Wahl aus dem Programm-Service-Angebot der Zeitschriften

PC Magazin	Happy-Computer-Sonderheft	Computer persönlich
PC Magazin Plus	Amiga-Magazin	64'er-Magazin
Happy-Computer	Amiga-Sonderheft	64'er-Sonderheft

bestellen – egal, ob diese DM 29,90 oder DM 34,90 kosten. Das Scheckheft können Sie per Verrechnungsscheck oder mit der eingehafteten Zahlkarte direkt beim Verlag bestellen.

Kennwort: Software-Scheckheft, Bestell-Nr. 39100.

ANWENDUNGEN

die Kursdatei (etwa Ihren Vornamen) eingeben. Wenn Sie bei den ersten Versuchen schon eine Datei unter diesem Namen erstellt haben, geben Sie drei Sterne und ein Leerzeichen vor dem eigentlichen Namen ein, so daß in jedem Fall eine völlig neue Kursdatei angelegt wird. Nun sehen Sie sich die aktuellen Voreinstellungen des Programms mit <F7> an. Ändern sollten Sie zunächst allerdings nur das Default-Laufwerk, um es Ihren Gegebenheiten anzupassen. Gehen Sie zurück ins Hauptmenü und drücken Sie <F1>. Dort geben Sie »1« gefolgt von <RETURN> ein. Nach dem Ladevorgang können Sie die Lektion starten. Bitte sehen Sie sich aber vorher die folgenden zehn Grundregeln (das sollten Sie zur besseren Einprägung möglichst vor jedem Start einer Lektion tun) an:

1. Arbeiten Sie möglichst regelmäßig mit dem Programm.
2. Bearbeiten Sie an einem Tag nur eine Lektion.
3. Wiederholen Sie jede Lektion am besten mindestens einmal.
4. Lassen Sie sich nicht von anfänglichen Schwierigkeiten entmutigen.
5. Greifen Sie die Grundstellung jedesmal blind.

6. Sehen Sie **nie** auf die Tastatur, orientieren Sie sich an der Hilfstastatur am Bildschirm.

7. Der Anschlag erfolgt kurz und ruckartig aus dem Muskel heraus.

8. Gehen Sie nach jedem Anschlag schnell in die Grundstellung zurück.

9. Tippen Sie langsam, aber gleichmäßig im Takt.

10. Halten Sie die Handgelenke möglichst hoch. Die Hand darf nicht flach auf der Tastatur liegen, sie muß angewinkelt sein.

11. Es kann nicht schaden, wenn Sie Ihre Finger mit einer speziellen Gymnastik trainieren.

Noch ein Hinweis zu Lektion 12: Beachten Sie, daß eine zu »shiftende« Taste, die sich im rechten Teil der Tastatur befindet, immer zusammen mit der linken SHIFT-Taste (und umgekehrt) angeschlagen wird. Das Programm unterstützt Sie dabei, indem immer die richtige zu drückende SHIFT-Taste hervorgehoben wird.

Haben Sie sich all dies zu Herzen genommen, kann es losgehen: Drücken Sie <F2> und tippen Sie nun alle Zeilen der Lektion ein. Schließen Sie jede Zeile mit <RETURN> ab. Werden Sie nicht bei jedem Fehler nervös. Errare huma-

Linke Hand:	a	kleiner Finger
	s	Ringfinger
	d	Mittelfinger
	f	Zeigefinger
Rechte Hand:	j	Zeigefinger
	k	Mittelfinger
	l	Ringfinger
	ö	kleiner Finger

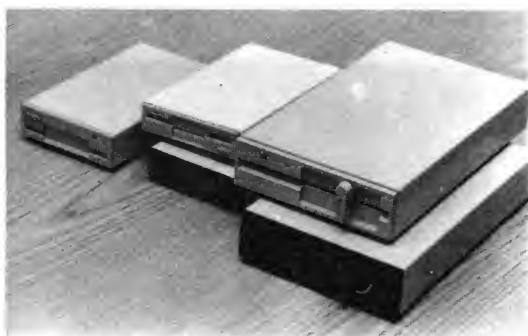
Tabelle 1. Die Grundstellung beim Maschineschreiben ist die Basis für alle Übungen

num est! (lat.: Irren ist menschlich!) Bleiben Sie konzentriert und denken Sie daran, daß die ersten Lektionen die schwierigsten sind.

Sind Sie mit der Lektion fertig, werden Ihnen die Ergebnisse gezeigt. Hinter Fehler, Fehlerquotient und Anschläge pro Minute stehen je zwei Zahlen, jedoch nur dann, wenn Sie die Lektion insgesamt mindestens zweimal bearbeitet haben. Die zweite Zahl stellt jeweils die Differenz zum Ergebnis des letzten Mals dar. Die Anzeige der Anschläge je Minute sollten Sie bei den ersten Lektionen einfach übersehen. Sie sind beim Lernen völlig unwichtig und werden in der Punktwertung kaum berücksichtigt. Wichtig ist, daß Sie möglichst gleichmäßig tippen

und wenige Fehler machen. Wenn Sie nun vom Hauptmenü mit <F6> die Auswertung aufrufen, können Sie sich die erreichte Punktzahl ansehen. Sie sollten diese jedoch nicht mit Schulnoten vergleichen. Denken Sie daran, daß Sie hier völlig ohne Druck lernen. Die Punkte und auch die Tippgeschwindigkeit sollen Ihnen vor allem als Vergleich dienen, wenn Sie schon einige Lektionen hinter sich haben. Punktwerte unter »50« sind jedoch ein Warnzeichen dafür, daß Sie diese Lektion wiederholen müssen. Wenn Sie mehr als »80« Punkte erreichen, können Sie sehr zufrieden sein, Werte über »100« oder gar über »120« sind sehr gut.

(Robert Kretzschmar/
M. Jobst/rs)



Profilaufwerke für Ihren AMIGA!

2 Jahre Garantie, 14 Tage Umtauschrecht, professionelle Leiterplatten, fast alle ICs gesockelt, Bedienungsanleitung, auf Wunsch vollständiges Manual mit allen Daten zu den Laufwerken lieferbar, 2tägiger Liefer-Rhythmus.

Für alle Laufwerke gilt:

- voll kompatibel zur vorhandenen Soft- und Hardware,
- komplett anschlussfertig,
- amigafarbenes Metallgehäuse,
- abschaltbar (intelligente Abschaltung),
- Kapazität 880 KB,
- korrekte LED-Ansteuerung,
- erkennen Disk-Change,
- kein separates Netzteil nötig (Stromversorgung über AMIGA)
- an alle AMIGA-Modelle anschließbar.

Für unsere 5.25"-Laufwerke gilt zusätzlich:

- TEAC-Laufwerke auf Wunsch umschaltbar 40/80 Tracks

Alle Laufwerke sind auch mit Busdurchführung lieferbar und sind dann mit einer automatischen Laufwerkserkennung ausgestattet, so daß beim Anschluß eines weiteren Laufwerkes an unser Laufwerk das Fremdlaufwerk auf die nächsthöhere Laufwerksadresse als unser Laufwerk gesetzt wird. Aufpreis: 25,- DM

SDN 3.5" - 1037A 249,-

- Superslimline, nur 25,4 mm hoch
- nur noch 5V Spannungsversorgung
- sehr niedriger Stromverbrauch

SDN 3.5" Digital - 1037A 289,-

- durchgeführter Bus bis df3: mit automat. Laufwerkserkennung
- Digitale Trackanzeige mit Helligkeitsregulierung

SDN 5.25" - TEAC FD 55 FR 299,-

- schwarze Frontblende
- unformatiert 1 MB Kapazität

SDN 5.25" - NEC1157C 299,-

- helle Frontblende
- Diskettenauswurf durch Feder
- unformatiert 1,67 MB Kapazität

SDN 5.25" - TEAC FD 55 GFR 309,-

- helle Frontblende
- unformatiert 1,67 MB

SDN 5.25" Digital 339,-

- durchgeführter Bus bis df3: mit automat. Laufwerkserkennung
- Digitale Trackanzeige mit Helligkeitsregulierung

SONDERAKTION NUR SOLANGE VORRAT REICHT!

SDN 3.5" - 1036A 229,-

komplett anschlussfertig für alle Amigas

SDN 3.5" intern 199,-

- für Einbau in A2000
- komplett mit Einbauanleitung und Montagematerial
- helle Frontblende

Rohlaufwerke

(unmodifiziert, ohne Gehäuse u. Kabel):

NEC 1036A	195,-
NEC 1037A	195,-
NEC 1157C	229,-
TEAC FD 55 FR	229,-
TEAC FD 55 GFR	229,-
Gehäuse (NEC 1036, 1037)	19,-
Gehäuse (NEC 1157, TEAC FD 55)	22,-
AMIGA 2000 & 1084	2350,-
XT-Karte	890,-
NEC P2200	879,-
NEC P6	1199,-
Star LC10	649,-
Star LC10 Color	750,-
Mitsubishi EUM-1481A	1499,-
NEC Multisync II	1599,-
Festplatte 30 MB - 5.25"	849,-
- für A2000 intern	
Festplatte 20 MB - 3.5"	949,-
- für A2000 intern	
Festplatte 30 MB - 3.5"	1049,-
- für A2000 intern	
Festplatte 30 MB	949,-
- für A500/1000 extern	
OMTI-Controller 5520	179,-
OMTI-Controller 5527	199,-
SCSI/ST 506-Controller	449,-
ST 238 HD 30 MB	549,-
ST 225 HD 20 MB	529,-
SCSI-D5126 HD 25 MB	699,-
Bootselektor	19,-
Farbband NEC P6	17,-
Farbband NEC P2200	17,-

WIR FÜHREN GÜNSTIG UND SCHNELL REPARATUREN AN ALLEN AMIGA-MODELLEN AUS.

G

RAFIK-TOOLS

Bilder lernen laufen

Filmen ist ein tolles Hobby. Wußten Sie schon, daß Sie die Ausrüstung für Computer-Movies bereits besitzen? Sie benötigen nur noch diese Listings, um den »Videorecorder« in Ihrem Amiga zu starten. Zusätzlich erhalten Sie die nötigen Basis-Informationen zur Animation von IFF-Bildern.

Vielleicht kennen Sie die Programme »Videoscape 3D« oder »Animate 3D«. Mit diesen lassen sich Animationen recht gut erstellen. Wer allerdings selbst gemalten Einzelbildern zur Bewegung verhelfen will, ist mit unserem Projekt besser bedient.

Zum Animieren Ihrer selbst erstellten Bilder benötigen Sie nur drei Dinge:

1. Ein Malprogramm, das Ihrer Kreativität freien Lauf läßt und die so entstandenen Einzelbilder der Animation im IFF-ILBM-Format abspeichern kann (z.B. »Deluxe Paint II«, »Photon Paint«, »Digi Paint«).
2. »Play« und »Anim«. Dies sind die beiden Programme, mit denen Sie fortan Ihren Amiga zum »Videorecorder« umgestalten.
3. Etwas Geduld, da das Erzeugen der Animationsdateien mitunter viel Zeit in Anspruch nehmen kann.

Bevor wir Sie nun mit der Bedienung der Programme »Play« und »Anim« (Listing 1 bis 4) vertraut machen, vorab ein wenig Theorie zum Thema Animation.

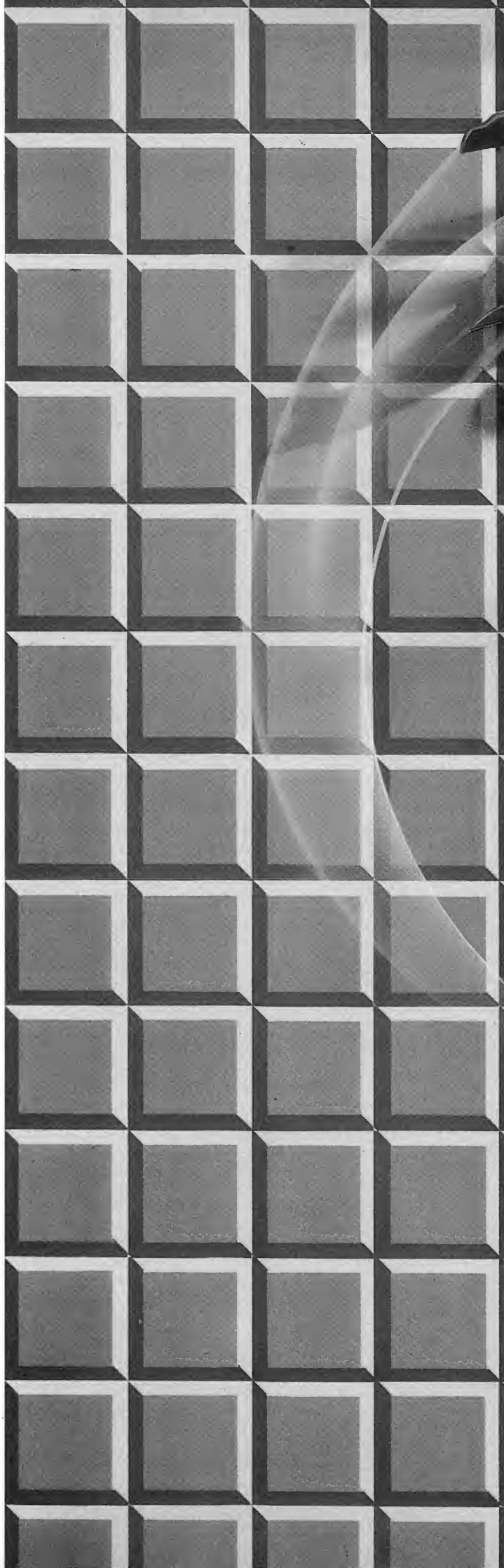
Eine Animation setzt sich aus einer bestimmten Anzahl von Einzelbildern zusammen. Die einfachste Möglichkeit diese in Bewegung zu setzen wä-

re, die Bilder der Reihe nach in den Bildschirmspeicher zu laden. Da hier ständig auf Diskette zugegriffen wird und zusätzlich die IFF-Bilder noch entpackt werden müßten, würde dieser Vorgang zu lange dau-

Von »Diashows« und »Page Flipping«

ern — die Animation ist dann nicht mehr fließend und hätte wohl doch eher das Attribut »Diashow« verdient.

Weiter könnte man sämtliche Bilder erstmal in den Speicher laden und gleichzeitig entpacken, um den Diskettenzugriff zu minimieren und Rechenzeit zu sparen. Diese Methode wird auch als »Page Flipping« bezeichnet und ist natürlich äußerst schnell, solange die Bilder im Chip-Memory liegen. Wenn man allerdings bedenkt, daß eine vier Bitplanes »tiefe« und 320 x 256 Pixel große LoRes-Grafik 40 KByte Speicher benötigt, sind die Animationen bei einem Amiga mit 512 KByte Speicher leider nur sehr kurz. Es wird also sehr viel Speicher vorausgesetzt. Bei längeren Filmen müßte das Fast-Memory benutzt werden, auf das aber von den Custom-





chips nicht zugegriffen werden kann. Demnach können die Bitmaps auch nicht im Fast-Memory liegen — die Bilder müßten per Prozessor zuerst »herunter kopiert« werden. »Page Flipping« sollte dann vielleicht eher als »Page Copying« bezeichnet werden. Eine längere Animation wäre aber trotz des Kopieraufwandes noch fließend. Bei einem Rechner mit großem Speicherplatz ist das »Page Flipping« daher noch zu vertreten.

Das nächste Bild bitte

Werden nun zwei aufeinanderfolgende Einzelbilder eines Films verglichen, stellt man oft fest, daß diese sich sehr ähnlich sind. Änderungen von Bild zu Bild betreffen demnach nur bestimmte Teile der Grafik. Es ist daher naheliegend, sich bei einer Animation nicht jedes Bild, sondern nur dessen Unterschied zu seinem Vorgänger zu merken.

Eine Ausnahme ist natürlich das erste Bild der Animation. Dieses hat keinen Vorgänger und muß daher vollständig vorhanden sein.

Das zuletzt beschriebene Verfahren nennt man »Delta-Animation«.

Das IFF-ANIM-File

Wir wissen jetzt, wie unser Projekt optimal arbeitet. In diesem Moment greift unser Programm »Anim« ein: Es verknüpft sozusagen die Bilderkette zu einem einzigen File. Die entstandene Datei hält sich dabei an den von Aegis definierten IFF-ANIM-Standard. Wie bei IFF-Dateien üblich, werden die Informationen in entsprechenden »Chunks« untergebracht. Die Profis kennen diesen Begriff sicher und wissen auch, was dahintersteckt. Für die weniger erfahrenen Amiga-Anwender sei hier nur erklärt, daß Chunks Standard-Informationsblöcke sind, mit denen Grafiken untereinander kompatibel bleiben. So existieren für die Differenzdaten der Bilder zwei Chunks, wobei der eine (DELTA-Chunk) die eigentlichen Daten enthält und der andere (ANHD) Aufschluß über das verwendete Kompressionsverfahren gibt. »Anim« erledigt all dies von alleine, so daß Sie sich nicht um lästige Details kümmern müssen.

Fließende Bewegungen durch »Double Buffering...«

Beim Abspielen eines Films werden normalerweise zwei Bitmaps zur Darstellung der bewegten Grafik verwendet. Die eine wird zum Anzeigen auf dem Bildschirm und die andere zum Aufbauen des neuen Bildes im Hintergrund verwendet. Man verhindert so Störungen, die beim gleichzeitigen Verändern und Anzeigen einer Grafik entstehen.

...und »interleave 2« zum schnelleren Abspielen

Nun wird es ein wenig kompliziert. Um das Abspielen einer Animationsdatei zu verstehen, sollten Sie immer folgende Grundsätze beachten:

1. Die Displaybitmap ist immer die, die gerade angezeigt wird.
2. Die Arbeitsbitmap ist diejenige, die gerade nicht angezeigt wird. Diese Bitmap kann also manipuliert werden (Double Buffering).
3. Soll das Folgebild anhand der Differenzdaten korrekt ermittelt werden, so muß in der Arbeitsbitmap genau die erste der beiden Grafiken vorliegen, die auch vorlagen, als die Differenzdaten erzeugt wurden. Nur so entspricht das Folgebild dem Bild, das beim Vergleich das zweite war.
4. Das Umschalten von einem Bild zum nächsten kann nur innerhalb der vertikalen Austastlücke geschehen.
5. Nach dem Umschalten wird die Displaybitmap zur Arbeitsbitmap und umgekehrt.

Versuchen wir einmal, den Abspielvorgang nachzuvollziehen. Es sei gerade ein Folgebild in der Arbeitsbitmap fertiggestellt. Demnach ist dieses Bild von der Reihenfolge her das Letzte. Wäre die Bilderkette durchnummeriert, so hätte es eine höhere Nummer, nennen wir sie einfach »n«. Nun wartet unser Amiga auf die Austastlücke. Nach deren Erreichen wird die Arbeitsbitmap zur Displaybitmap. Die alte Displaybitmap, deren Grafik nach Erreichen der Austastlücke mindestens einmal vom Monitor dargestellt wurde, wird nun zur neuen Arbeitsbitmap. Das Bild in unserer jetzigen Ar-

beitsbitmap hat demnach auch eine niedrigere Nummer, also n-1.

Wir möchten nun in dieser Arbeitsbitmap das nächste Bild erzeugen, dessen Nummer ja n+1 sein muß, da n sich gerade in der Displaybitmap befindet und demnach schon fertiggestellt ist. Gehen wir davon aus, daß bei der Bildung der Differenzdaten zwei direkt einander folgende Bilder dienten, also Bild n und Bild n+1 oder Bild n-1 und n, so muß sich nach Grundsatz 3 in unserer momentanen Arbeitsbitmap Bild n befinden, um mit den kommenden Differenzdaten Bild n+1 zu erzeugen. Es liegt aber nun Bild n-1 vor. Jetzt gibt es zwei Möglichkeiten: Entweder wird Bild n aus der Displaybitmap in die Arbeitsbitmap kopiert oder aber die Differenzdaten werden aus Bildern gebildet, die um zwei Einzelbilder auseinanderliegen (z.B. Bild n-2 und n oder Bild n-1 und n+1). Das Auseinanderliegen der Einzelbilder wird auch als Interleave bezeichnet. In den IFF-ANIM-Files verwendet man im allgemeinen einen Interleave-Faktor von 2, da somit ein Kopiervorgang wegfällt. Eine Ausnahme stellen Bild 1 und 2 dar, da Bild 1, wie schon erwähnt, als vollständige Grafik vorhanden sein muß und für das zweite noch kein um zwei Einzelbilder zurückliegendes Bild vorhanden ist.

Der CONTINUOUS-Play-Mode

Bei manchen Animationen kommt es vor, daß das erste und letzte Bild identisch sind. In diesem Fall ist es nicht mehr notwendig, die Anfangsgrafik vollkommen neu aufzubauen, wie es beim Start der Animation notwendig war. Diese liegt nun in identischer Form sowie so im Speicher vor. Mit anderen Worten: Bei einem zweiten Durchlauf erhalten wir nur dann einen fließenden Übergang, wenn Start- und Endgrafik vollkommen übereinstimmen. Ist diese Bedingung nicht gegeben, so wird die Animation als »NON-CONTINUOUS« bezeichnet.

Wie oben schon erläutert, wird im IFF-ANIM-Standard ein Interleave-Faktor von 2 benutzt. Konsequenterweise genügt es dann nicht, für eine CONTINUOUS-Animation nur Start- und Endbild gleich zu lassen, sondern es müssen die

ersten und letzten beiden Bilder identisch sein, da beim Start unseres Films ja bereits zwei Bilder als Grafikunterlage dienen (liegen n Bilder vor, so müssen 1 und n-1 sowie 2 und n übereinstimmen).

Die Anwendung: »ANIM«

Unsere beiden Programme »Play« und »Anim« können nur vom CLI aus bedient werden. Am besten kopieren Sie diese Tools in den c-Ordner oder in das aktuelle Verzeichnis. Wie schon erwähnt, dient »Anim« zur Erstellung eines IFF-Animationsfiles. Ein Aufruf könnte so aussehen:

```
anim df0:bild df1:
output.anim
```

Der Parameter »df0:bild« bezeichnet die Bilderkette. Nach Programmstart versucht »Anim«, das File »df0:bild1.pic« zu öffnen. Ist die Suche erfolgreich verlaufen, erscheint ein Screen und die Bilddaten werden eingelesen und angezeigt.

Anschließend wird nacheinander auf die Bilder »df0:bild 2.pic«, »df0:bild3.pic« bis zum letzten vorhandenen Bild der Kette zugegriffen.

Gleichzeitig werden die dabei anfallenden Differenzdaten in das File »df1:output.anim« geschrieben. Sollten die Regeln des IFF-Standards beim Einlesen eines Bildes nicht beachtet worden sein oder sind sonstige Fehler aufgetreten, wie zu wenig Speicher, wird die Routine abgebrochen.

Leider ist eine Diskette mit 880 KByte schnell voll — besonders, wenn die Animation aus Hires-Grafiken besteht. Um diesen Engpaß zu umgehen, kann »Anim« noch während des Arbeitens mit einem Malprogramm im Multitasking-Betrieb aufgerufen werden. Geben Sie einfach »run anim« vom CLI aus ein, nachdem das Malprogramm Ihrer Wahl bereits gestartet wurde. Es wird dann ein Fenster geöffnet, mit dessen Hilfe Sie sich einen Screen aus der Liste der gerade vorhandenen aussuchen können. Im Normalfall wird dies der Screen Ihres Malprogrammes sein. Anhand der gezeigten Parameter können Sie leicht sehen, welches der richtige Grafikhintergrund ist. Das »First«-Gadget zeigt den ersten, das »Next«-Gadget den nächsten Screen der Liste an. Mit »OK« bestätigt man schließlich seine Wahl

und es wird ein weiteres Fenster geöffnet. Dieses Fenster ermöglicht die Erstellung des Animationsfiles und »klaut« sich die Grafiken sozusagen aus dem gerade ausgewählten Screen.

Da dieses Fenster im Workbench-Screen geöffnet wird, muß immer zwischen diesem und Ihrem Arbeitsscreen umgeschaltet werden. Dieses erreichen Sie am einfachsten mit <Amiga-N> und <Amiga-M>.

Erstellung mit Deluxe Paint

Nehmen wir an, bei Ihrem Malprogramm handelt es sich um »Deluxe Paint II«. Entwerfen Sie nach Belieben das erste Bild des Films. Klicken Sie nun das »Save Next Frame«-Gadget an, um das erste Einzelbild zu speichern. Danach erscheint ein File-Requester, der die Auswahl des zu erzeugenden Animationsfiles ermöglicht. Ist dies geschehen,

werden die Grafikdaten in dieses File geschrieben. In der Titelzeile unseres Fensters erscheint dann »Working...Please wait.«. Sie sollten dann warten, bis der Amiga seine Arbeit erledigt hat. Nach jedem weiteren erstellten Einzelbild wird wieder »Save Next Frame« betätigt, so daß jeweils immer die Differenzdaten im Output-File untergebracht werden können. Sind Sie der Meinung, das letzte Bild gemalt zu haben, wird mit »Finish« der Vorgang beendet und unser Animationsfile geschlossen, was unter keinen Umständen vergessen werden sollte! Weiter sollten Sie zwei Punkte unbedingt beachten, um sich Enttäuschungen zu ersparen:

1. Es wird genau das gespeichert, was auf dem Arbeitsscreen zu sehen ist, bei »Deluxe Paint II« also möglicherweise auch die Titelzeile und die Bedienungs-Gadgets auf der rechten Seite. Diese sind mit <F10> auszublenden (dies gilt natürlich nicht unbedingt für andere Malprogramme). Auch ein Pinsel (brush), der

gerade in Gebrauch ist und mit der Maus auf der Arbeitsgrafik entlangfährt, kann möglicherweise mit im angelegten File (outputfile) landen.

2. Während in unserem Fenster die Meldung »Working...Please wait.« zu sehen ist, dürfen im Arbeitsscreen keinerlei Änderungen vorgenommen werden! Beachten Sie dies nicht, werden die Differenzdaten falsch ermittelt und die Chunks unseres Animationsfiles erhalten fehlerhafte Längen. Bei »DPaint II« müßte daher beispielsweise darauf geachtet werden, daß schon durch leichte Mausebewegungen, zum Beispiel bei einem aktivem Brush, keine Manipulation der Grafik entsteht. Dies gilt auch, wenn die Workbench im Vordergrund dargestellt wird!

Dieses Verfahren erscheint anfangs äußerst unkomfortabel. Wer aber jedes Bild speichern möchte, um es nachträglich noch verändern zu können, kommt bei langen Animationen um eine Festplatte nicht herum.

Spielerien mit »Play«

Ist der Film nun endlich in einer Datei untergebracht, können Sie mit unserem »Anim Player« das Ganze zum Leben erwecken. Nach dem oben erklärten Beispiel müßte »play df1:output.anim« eingegeben werden.

Der digitale Film wird dann in den Speicher eingelesen — sofern er sich an die IFF-Konventionen hält und genug Speicher vorhanden ist und einmal abgespielt. Wem das zuwenig ist, der kann mittels »play -L df1:output.anim« den Film in einer Endlosschleife laufen lassen. Das gilt auch bei der Eingabe von »play -C df1:output.anim«, nur sollte es sich bei »output.anim« dann um eine Continuous-Animation handeln (siehe oben).

Letztendlich kann der Trickfilm mit der linken Maustaste angehalten und mit der rechten unterbrochen werden, so daß Sie sich wieder im CLI befinden. (Christian Wolf/irs)

Sie benötigen zu den hier abgedruckten Listings noch das Programm »mydir.c«, das wir im Amiga-Sonderheft 1 bereits auf der Seite 63 veröffentlicht haben. Sollten Sie nicht im Besitz dieses Heftes sein, dann können Sie es bei der Redaktionsservice (siehe Impressum) anfordern. Auf der Programmservice-Diskette ist »mydir.c« enthalten.

Programmname:	anim.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztek-C V3.6
Aufrufe:	cc +l -s +idmp anim.c

Programmname: anim.c

```

1 CVO #define OK 1
2 a2 #define NEXT 3
3 r3 #define FIRST 4
4 7t #define PLAY 5
5 6f #define SAVE 6
6 GW #define FINISH 7
7 am #define BUFFERSIZE 4000
8 nr #define NextLong ReadLongWord(MyFile)
9 rh #define MAKE_ID(a,b,c,d) ( (long)(a)<<24 | (long)(b)<<16 | (c)<<8 | (d) )
10 QJ #define NOTITLE NULL
11 g7 #define FORM MAKE_ID('F', 'O', 'R', 'M')
12 6G #define ILBM MAKE_ID('I', 'L', 'B', 'M')
13 pc #define BMHD MAKE_ID('B', 'M', 'H', 'D')
14 fe #define CMAP MAKE_ID('C', 'M', 'A', 'P')
15 AP #define BODY MAKE_ID('B', 'O', 'D', 'Y')
16 Vv #define GRAB MAKE_ID('G', 'R', 'A', 'B')
17 ls #define DEST MAKE_ID('D', 'E', 'S', 'T')
18 df #define SPRT MAKE_ID('S', 'P', 'R', 'T')
19 xc #define CAMG MAKE_ID('C', 'A', 'M', 'G')
20 og #define CRNG MAKE_ID('C', 'R', 'N', 'G')
21 GP #define CCRT MAKE_ID('C', 'R', 'T')
22 Y6 #define DPPV MAKE_ID('D', 'P', 'P', 'V')
23 mL struct BitMapHeader
24 8V9 UWORD w, h;
25 q8 UWORD x, y;
26 Eu UBYTE Planes;
27 0a UBYTE masking;
28 1I UBYTE compression;
29 Lt UBYTE pad1;
30 PD UWORD transparentColor;
31 uZ UBYTE xAspect, yAspect;
32 Rn WORD pageWidth, pageHeight;
```

```

33 qm8 };
34 eP0 struct BitMapHeader BitMapHeader;
35 8p char ColorMap[192];
36 1c char *BodyChunk;
37 1I unsigned long BodySize;
38 HM struct NewScreen NewS1 = {
39 ss7 0,0,0,0,0,0,0,CUSTOMSCREEN,NULL,NOTITLE,0 };
40 5e0 struct Screen *MyScreen;
41 pv BOOL first_body=FALSE;
42 3t USHORT MyLines[] = { 0,0,69,0,69,11,0,11,0,0 };
43 1h USHORT PlayLines[] = { 0,0,119,0,119,11,0,11,0,0 };
44 0Z USHORT SaveLines[] = { 0,0,127,0,127,11,0,11,0,0 };
45 xb USHORT FinishLines[] = { 0,0,55,0,55,11,0,11,0,0 };
46 1L struct Border MyBorder = { 0,0,1,3,JAM2,5,&MyLines,NULL };
47 Ny struct Border PlayBorder = { 0,0,1,3,JAM2,5,&PlayLines,NULL };
48 MU struct Border SaveBorder = { 0,0,1,3,JAM2,5,&SaveLines,NULL };
49 EE struct Border FinishBorder = { 0,0,1,3,JAM2,5,&FinishLines,NULL };
50 bK struct IntuiText TextNext = { 1,0,JAM2,20,3,NULL,"Next",NULL };
51 ex struct IntuiText TextFirst = { 1,0,JAM2,16,3,NULL,"First",NULL };
52 Gw struct IntuiText TextOk = { 1,0,JAM2,28,3,NULL,"Ok",NULL };
53 z2 struct IntuiText TextPlay = { 1,0,JAM2,3,2,NULL,"Play Animation",NULL };
54 d0 struct IntuiText TextSave = { 1,0,JAM2,3,2,NULL,"Save Next Frame",NULL };
55 zB struct IntuiText TextFinish = { 1,0,JAM2,3,2,NULL,"Finish",NULL };
56 mX struct Gadget GCGNext = { NULL,85,50,70,12,GADGHCMP,RELVER,IFF,
```

Listing 1. »anim.c« erzeugt aus Grafiken bewegte Bilder. Bitte mit dem Checksummer (Seite 159) eingeben.


```

57 BRP          BOOLGADGET,&MyBorder,0,&TextNext,
                0,0,NEXT,0;
58 T00 struct Gadget GGFIRST = { &GGNext,7,50,70,12,GADGHCOMP,REL
VERIFY,
59 mKQ          BOOLGADGET,&MyBorder,0,&TextFirs
                t,0,0,FIRST,0;
60 Sm0 struct Gadget GGOK = { &GGFirst,163,50,70,12,GADGHCOMP,REL
VERIFY,
61 pVN          BOOLGADGET,&MyBorder,0,&TextOK,0,0,
                OK,0;
62 CX0 struct Gadget GGPlay = { NULL,7,12,120,12,GADGHCOMP,RELVER
IFY,
63 uNP          BOOLGADGET,&PlayBorder,0,&TextPla
                y,0,0,PLAY,0;
64 U80 struct Gadget GGSave = { &GGPlay,135,12,128,12,GADGHCOMP,R
ELVERIFY,
65 bnQ          BOOLGADGET,&SaveBorder,0,&TextSa
                ve,0,0,SAVE,0;
66 OH0 struct Gadget GGFinish = { &GGSave,271,12,56,12,GADGHCOMP,
RELVERIFY,
67 pXR          BOOLGADGET,&FinishBorder,0,&Tex
                tFinish,0,0,FINISH,0;
68 Rd0 char MyTitle[] = "Delta Anim 0.6 1988 by CWF";
69 FZ char Working[] = "Working...Please wait.";
70 LI struct NewWindow NewWindow = {
71 Y97          0,0,240,66,0,1,
72 N1          GADGETUP,
73 87          ACTIVATE|WINDOWDRAG|WINDOWDEPTH,
74 4C          &GGOK,NULL,
75 fF          "Select your Screen",
76 Qu          NULL,NULL,
77 vp          640,200,640,200,
78 KO          WBENCHSCREEN };
79 p50 struct NewWindow AnimWindow = {
80 d97          0,0,334,26,0,1,
81 1Q          GADGETUP|CLOSEWINDOW,
82 r2          ACTIVATE|WINDOWDRAG|WINDOWDEPTH|WINDOWCLOSE,
83 2h          &GGFinish,NULL,
84 3L          MyTitle,
85 Z3          NULL,NULL,
86 4y          640,200,640,200,
87 TX          WBENCHSCREEN };
88 dc0 struct Window *MyWindow;
89 uN struct IntuitionBase *IntuitionBase;
90 Er struct GfxBase *GfxBase;
91 nZ struct RastPort *ThisRastPort;
92 JO char MyBuffer[40];
93 DL char FullName[108],FileName[108];
94 Ij APTR Buffer;
95 n9 BOOL Resident=TRUE;
96 bQ char MyColorMap[3*64];
97 V5 FILE *MyFile;
98 vM unsigned long offset;
99 sZ unsigned long planes[8];
100 zU char Pic[5]=".pic",Number[4];
101 8d struct BitMap BackupBitMap1,BackupBitMap2;
102 SS struct BitMap *bbm1=&BackupBitMap1,*bbm2=&BackupBitMap2;
103 nm reverse(s)
104 m61 char s[];
105 ck char c;
106 2w int i,j;
107 s2 for (i=0,j=strlen(s)-1;i<j;i++,j--)
108 LJ7 c=s[i];
109 Zs s[i]=s[j];
110 HO s[j]=c;
111 Vm0 itoa(n,s)
112 uE1 char s[];
113 nz int n;
114 i3 int i,sign;
115 78 if ((sign = n) < 0) n=-n;
116 qB i=0;
117 uH do
118 tJ3 s[i++] = n%10+'0';
119 K52 } while ((n /= 10) > 0);
120 V71 if (sign < 0) s[i++]='-';
121 GY s[i]='\0';
122 y0 reverse(s);
123 5X0 ShowScreenData(ThisScreen)
124 LT1 struct Screen *ThisScreen;
125 wV char *ScreenName=ThisScreen->Title;
126 i1 UWORD Modes=ThisScreen->ViewPort.Modes;
127 yZ SetDrMd(ThisRastPort,JAM2);
128 7t SetAPen(ThisRastPort,0);

```

```

129 E2 RectFill(ThisRastPort,2,10,237,45);
130 1c SetDrMd(ThisRastPort,JAM2);
131 DO SetAPen(ThisRastPort,1);
132 Me Move(ThisRastPort,7,20);
133 b3 Text(ThisRastPort,"Res: ",5);
134 r1 itoa(ThisScreen->Width,MyBuffer);
135 IU Text(ThisRastPort,MyBuffer,strlen(MyBuffer));
136 uJ Text(ThisRastPort," x ",3);
137 uk itoa(ThisScreen->Height,MyBuffer);
138 LX Text(ThisRastPort,MyBuffer,strlen(MyBuffer));
139 qJ Move(ThisRastPort,130,20);
140 Oa Text(ThisRastPort,"Depth: ",7);
141 Bd itoa(ThisScreen->RastPort.BitMap->Depth,MyBuffer);
142 Pb Text(ThisRastPort,MyBuffer,strlen(MyBuffer));
143 f0 Move(ThisRastPort,7,32);
144 Zg Text(ThisRastPort,"Title: ",7);
145 1E if (ScreenName) Text(ThisRastPort,ScreenName,strlen(Scre
nName));
146 hX else Text(ThisRastPort,"untitled",8);
147 rF Move(ThisRastPort,7,44);
148 sg Text(ThisRastPort,"Mode: ",6);
149 4C if (Modes & HIRES) Text(ThisRastPort,"HIRES",5);
150 XC else if (Modes & HAM) Text(ThisRastPort,"HAM",3);
151 hW else if (Modes & EXTRA_HALFBRITE) Text(ThisRastPort,"EXTR
A_HALFBRITE",15);
152 fP else Text(ThisRastPort,"LORES",5);
153 4P if (Modes & LACE) Text(ThisRastPort,"] INTERLACE",10);
154 A40 struct Screen *GetScreen()
155 iW1 ULONG Class;
156 7A struct IntuiMessage *MyMessage;
157 yW struct Screen *CurrentScreen=IntuitionBase->FirstScreen;
158 yx if (!(MyWindow=OpenWindow(&NewWindow))) return FALSE;
159 Tp ThisRastPort=MyWindow->RPort;
160 At ShowScreenData(CurrentScreen);
161 9q FOREVER
162 9J3 WaitPort(MyWindow->UserPort);
163 Ik if (MyMessage=GetMsg(MyWindow->UserPort))
164 oM8 Class=MyMessage->Class;
165 Bq switch(Class)
166 FhA case GADGETUP:
167 An switch(MyMessage->IAddress->GadgetID)
168 DfC case OK:
169 yA CloseWindow(MyWindow);
170 uk return(CurrentScreen);
171 aZ case NEXT:
172 GN if (CurrentScreen->NextScreen)
173 LFG ShowScreenData(CurrentScreen=CurrentScreen
->NextScreen);
174 jsC break;
175 T1 case FIRST:
176 L4 ShowScreenData(CurrentScreen=IntuitionBase->F
irstScreen);
177 mv break;
178 aU default:
179 ox break;
180 pyA break;
181 dX default:
182 r0 break;
183 PV3 ReplyMsg(MyMessage);
184 Gx0 CopyBitMap(s,d)
185 OC1 struct BitMap *s,*d;
186 tw char p=s->Depth,j=0;
187 qW register unsigned long c;
188 VQ unsigned long size = s->BytesPerRow * s->Rows;
189 gl register char *sp,*dp;
190 1T while (p--)
191 8w9 c=size;
192 FR sp=s->Planes[j];
193 u0 dp=d->Planes[j++];
194 i1 while (c--) *dp++=*sp++;
195 vH0 MakeColorMap(MyScreen)
196 bA1 struct Screen *MyScreen;
197 BA UWORD *cp=MyScreen->ViewPort.ColorMap->ColorTable;
198 oN char *cd=MyColorMap;
199 JI UWORD Count=MyScreen->ViewPort.ColorMap->Count,c;
200 JY for (c=0;c<Count;c++)
201 9P7 *cd++ = (*cp>>4) & 0xf0;
202 oW *cd++ = *cp & 0xf0;
203 mw *cd++ = *cp++ << 4;
204 Ca0 LoadColorMap(ThisView)
205 y71 struct ViewPort *ThisView;
206 3v char *c=ColorMap;
207 4T unsigned short i,d=1<<BitMapHeader.Planes;

```



```

208 VK      unsigned short r,g,b;
209 Eq      for (i=0;i<d;i++)
210 WZ7      r=c++>>4;
211 B3      g=c++>>4;
212 2p      b=c++>>4;
213 TU      SetRGB4(ThisView,i,r,g,b);
214 md0     BOOL MakeBackupRaster(bm,dbm)
215 6G1     struct BitMap *bm,*dbm;
216 c7      unsigned short p=bm->Depth,bpr=bm->BytesPerRow,r=bm->R
ows;
217 Kg      unsigned short planesize=bpr*r;
218 yw      char j=0;
219 mT      APTR PlanePtr;
220 Kx      InitBitMap(dbm,p,bpr*8,r);
221 Wy      while (p--)
222 Q09      if (PlanePtr=AllocMem(planesize,MEMF_PUBLIC))
223 yME      dbm->Planes[j++]=PlanePtr;
224 6S9      else return(FALSE);
225 Mk1      return TRUE;
226 re0     FreeBackupRaster(bm)
227 HA1     struct BitMap *bm;
228 oJ      unsigned short p=bm->Depth,bpr=bm->BytesPerRow,r=bm->R
ows;
229 Ws      unsigned short planesize=bpr*r;
230 A8      char j=0;
231 yf      APTR PlanePtr;
232 H1      while (p--) if (PlanePtr=bm->Planes[j++]) FreeMem(PlaneP
tr,planesize);
233 3s0     MakeAnim(ThisScreen)
234 7F1     struct Screen *ThisScreen;
235 0o      ULONG Class;
236 PS      struct IntuiMessage *MyMessage;
237 OW      struct BitMap *ThisBitMap=ThisScreen->RastPort.BitMap,*S
wap;
238 1S      BOOL FirstTime;
239 JT      if (!ThisScreen) return;
240 n3      if (!MyWindow=OpenWindow(&AnimWindow))) return;
241 Uo      if (!MakeBackupRaster(ThisBitMap,bbm1))
242 Xc6      FreeBackupRaster(bbm1);
243 AM      CloseWindow(MyWindow);
244 xW      return;
245 bw1     if (!MakeBackupRaster(ThisBitMap,bbm2))
246 bg6     FreeBackupRaster(bbm1);
247 ek      FreeBackupRaster(bbm2);
248 FR      CloseWindow(MyWindow);
249 2b      return;
250 aH1     FOREVER
251 aA3      WaitPort(MyWindow->UserPort);
252 jB      if (MyMessage=GetMsg(MyWindow->UserPort))
253 Fn8      Class=MyMessage->Class;
254 cH      switch(Class)
255 g8A      case GADGETUP:
256 bE      switch(MyMessage->IAddress->GadgetID)
257 LNC      case FINISH:
258 qa      if (MyFile) CloseAnim(FullName);
259 6F      break;
260 LB      case PLAY:
261 Un      if (MyFile) break;
262 G6      if (!GetFileName(File,FullName,"Select Pla
y File",Buffer,BUFFERSIZE,Resident)) break;
263 Xr      MyFile=0;
264 BK      break;
265 jS      case SAVE:
266 tZ      if (!MyFile)
267 eiH      if (!GetFileName(File,FullName,"Save
Animation",Buffer,BUFFERSIZE,Resident)) br
eak;
268 ch      SetWindowTitles(MyWindow,Working,0);
269 r2      if (!OpenAnim(FullName)) break;
270 cK      MakeColorMap(ThisScreen);
271 U4      CopyBitMap(ThisBitMap,bbm1);
272 X8      CopyBitMap(ThisBitMap,bbm2);
273 Cr      offset += write_ilbm(MyFile,MyColorMap,bb
m1,ThisScreen->ViewPort.Modes,TRUE);
274 gT      SetWindowTitles(MyWindow,MyTitle,0);
275 jr      FirstTime=TRUE;
276 5NC      else if (FirstTime)
277 lqH      SetWindowTitles(MyWindow,Working,0);
278 nF      offset += write_delta(MyFile,bbm1,ThisBitM
ap);
279 cC      CopyBitMap(ThisBitMap,bbm1);
280 mZ      SetWindowTitles(MyWindow,MyTitle,0);
281 BR      FirstTime=FALSE;

```

```

282 NAC      else
283 rWH      SetWindowTitles(MyWindow,Working,0);
284 wP      offset += write_delta(MyFile,bbm2,ThisBitM
ap);
285 kL      CopyBitMap(ThisBitMap,bbm2);
286 Ht      Swap=bbm2;
287 S9      bbm2=bbm1;
288 Ep      bbm1=Swap;
289 v1      SetWindowTitles(MyWindow,MyTitle,0);
290 bkC      break;
291 PJ      default:
292 dm      break;
293 enA      break;
294 AM      case CLOSEWINDOW:
295 RB      if (MyFile) CloseAnim(FullName);
296 PU      FreeBackupRaster(bbm1);
297 SY      FreeBackupRaster(bbm2);
298 3F      CloseWindow(MyWindow);
299 qP      return;
300 lu      break;
301 ZT      default:
302 nw      break;
303 LR3      ReplyMsg(MyMessage);
304 ry0     BOOL InitDisplay(bmh)
305 pF1     register struct BitMapHeader *bmh;
306 nI      if (bmh->h > 256) NewS1.ViewModes |= LACE;
307 xM      if (bmh->w > 352) NewS1.ViewModes |= HIRES;
308 C5      NewS1.ViewModes |= VP_HIDE;
309 RD      NewS1.Width=bmh->w;
310 83      NewS1.Height=bmh->h;
311 N5      NewS1.Depth=bmh->Planes;
312 HJ      if (!MyScreen=OpenScreen(&NewS1)) return FALSE;
313 Pa      if (!MakeBackupRaster(MyScreen->RastPort.BitMap,bbm1)) r
eturn FALSE;
314 Tf      if (!MakeBackupRaster(MyScreen->RastPort.BitMap,bbm2)) r
eturn FALSE;
315 TDO     WriteBody(bmh)
316 Pr1     struct BitMapHeader *bmh;
317 1U      char *(*PlanePtr)[8]=&MyScreen->BitMap.Planes[0];
318 IQ      char **CurrentPtr,*ReadPointer,*pointer;
319 4u      unsigned short h=bmh->h,bpl=bmh->w>>3,d=bmh->Planes,
PlaneOffset=0;
320 Ua      unsigned short j,count;
321 aV      char n,x;
322 Og      #asm
323 vkD      clr.l    d0
324 yo      clr.l    d1
325 1s      clr.l    d2
326 3p      move.l    _BodyChunk,a0 ;a0 = Zeiger auf Dat
en
327 vu      move.l    -4(a5),a1 ;Zeiger auf Planes[0
]
328 VJ      move.w    -18(a5),d0 ; d0 = Anzahl der Zeilen
329 oe      move.w    -20(a5),d1 ; d1 = Bytes pro Zeile
330 4s      move.w    -22(a5),d2 ; d2 = Anzahl der Planes
331 m7      clr.l    d4 ; d4=PlaneOffset
332 1z      clr.l    d6 ; d6=n=0
333 R7      clr.l    d5 ; PlaneOffset=0
334 QD      clr.l    d3 ;
335 Kk      clr.l    d7
336 Tn      bra      hstart
337 zx0     hloop:   move.w    d2,d3
338 9AD      move.l    a1,a2 ;a2=CurrentPtr
339 Ac      bra      pstart
340 r70     ploop:   move.l    (a2)+,a3 ;a3=pointer
341 2FD      add.l    d4,a3 ; += PlaneOffset
342 BY      clr.b    d5 ; count=0
343 pW0     cloop:   clr.w    d6
344 GND      move.b    (a0)+,d6 ;n=ReadPointer++
345 qX      tst.b    d6
346 5E      bmi      else
347 kv      addq.b    #1,d6 ;n++
348 0x      add.b    d6,d5 ;count += n
349 kk      bra      cpstart
350 je0     cploop:  move.b    (a0)+,(a3)+
351 nC      cpstart:  dbra     d6,cploop
352 KZD      bra      cstart
353 WVO     else:    move.b    (a0)+,d7 ;x=ReadPointer++
354 3cD      neg.b    d6 ;n = -n
355 s3      addq.b    #1,d6 ;n++
356 85      add.b    d6,d5 ;count += n

```

Listing 1. »anim.c« (Fortsetzung)


```

357 JU          bra    xstart
358 oB0 xloop:   move.b  d7,(a3)+
359 aY xstart:   dbra   d6,xloop
360 X3 cstart:   cmp.b   d1,d5
361 dxD          bne.s   cloop
362 iw0 pstart:   dbra   d3,ploop
363 MAD          add.w   d1,d4      ;PlaneOffset+=bpl
364 W50 hstart:   dbra   d0,hloop
365 XR1          #endasm
366 BV0 unsigned long ReadLongWord(MyFile)
367 Vq1 unsigned long MyFile;
368 Ns unsigned long Buffer;
369 fY Read(MyFile,&Buffer,4);
370 Z6 return Buffer;
371 LZ0 BOOL ReadILBM(MyFile)
372 av1 unsigned long MyFile;
373 KZ unsigned long ID,Length,Offset,End;
374 dh BOOL Error=FALSE;
375 cZ if ((ID=NextLong) == FORM)
376 KT6 End=NextLong-4;
377 JI if ((ID=NextLong) != ILBM)
378 gGB printf("Sorry, but this is not an IFF-FORM-File
...\\n");
Error=TRUE;
379 HW else
380 xk1 Error=TRUE;
381 JY6 printf("Sorry, but this is not anim IFF-ILBM-File...
\\n");
382 Rs Offset=0;
while (Offset < End && Error==FALSE )
383 Ii1 ID=NextLong;
384 Oo Length=NextLong;
385 n63 if (Length & 1) Length++;
386 cg Offset+=Length+8;
387 t8 switch (ID)
388 iu case BMHD:
389 3m Read(MyFile,&BitMapHeader,sizeof(BitMapHeader));
390 nC5 break;
391 Gk case CMAP:
392 FO Read(MyFile,ColorMap,Length);
393 Ot break;
394 UK case CAMG:
395 IR NewS1.ViewModes=NextLong;
396 uG break;
397 4e case BODY:
398 LU BodySize=Length;
399 TB if (!first_body)
400 pv if (!InitDisplay(&BitMapHeader))
401 ZL Error=TRUE;
402 psA break;
403 fuF LoadColorMap(&MyScreen->ViewPort);
404 Ra first_body=TRUE;
405 BKA if (!(BodyChunk=AllocMem(Length,MEMF_PUBLIC))) Error=
406 HY TRUE;
407 eK5 else
408 PC Read(MyFile,BodyChunk,Length);
409 JPA WriteBody(&BitMapHeader);
410 IG FreeMem(BodyChunk,Length);
411 uZ break;
412 Z15 case GRAB:
413 xN case DEST:
414 rb case SPRT:
415 py case CCRT:
416 hN case CRNG:
417 8m case DPPV:
418 QK Seek(MyFile,Length,0);
419 Ld break;
420 hq default:
421 VP printf("Sorry, but this is a corrupted IFF-File\\n");
422 L4 Error=TRUE;
423 zE break;
424 lu return Error;
425 lY1 void MakeFileName(FileName,count)
426 d10 char *FileName;
427 nX1 unsigned short count;
428 mZ strcpy(FullName,FileName);
429 QZ itoa(count,Number);
430 OD strcat(FullName,Number);
431 qQ strcat(FullName,Pic);
432 m8 printf("%s\\n",FullName);
433 GD BOOL Read_Files(SourceName,DestName)
434 kLO char *SourceName,*DestName;
435 Hu1 ULONG SourceFile,DestFile;
436 bP

```

```

437 48 UWORD c=1;
438 pg BOOL FirstTime=FALSE,SecondTime=FALSE;
439 DB struct BitMap *Swap;
440 BC while (MakeFileName(SourceName,c++),SourceFile=Open(FullN
ame,MODE_OLDFILE))
441 ZV9 if (ReadILBM(SourceFile))
442 ceE Close(SourceFile);
443 4D break;
444 8K9 if (!FirstTime)
445 v4E if (!OpenAnim(DestName))
446 fwJ printf("Can't open destination file...\\
n");
break;
447 8H MakeColorMap(MyScreen);
448 kxE CopyBitMap(MyScreen->RastPort.BitMap,bbm1);
449 yR CopyBitMap(MyScreen->RastPort.BitMap,bbm2);
450 lV offset += write_ilbm(MyFile,MyColorMap,bbm1,
MyScreen->ViewPort.Modes,TRUE);
451 dk FirstTime=TRUE;
452 p1 else if (!SecondTime)
453 Ru9 offset += write_delta(MyFile,bbm1,MyScreen->
RastPort.BitMap);
454 huE CopyBitMap(MyScreen->RastPort.BitMap,bbm1);
455 4X SecondTime=TRUE;
456 IT else
457 Cz9 offset += write_delta(MyFile,bbm2,MyScreen->
RastPort.BitMap);
458 o2E CopyBitMap(MyScreen->RastPort.BitMap,bbm2);
Swap=bbm2;
459 Ae bbm2=bbm1;
460 5h bbm1=Swap;
461 Gx Close(SourceFile);
462 2d if (MyFile) CloseAnim(DestName);
463 xz9 FreeBackupRaster(bbm1);
464 go1 FreeBackupRaster(bbm2);
465 8D if (MyScreen) CloseScreen(MyScreen);
466 BH main(argc,argv)
467 Gy int argc;
468 cy0 char **argv;
469 zZ1 IntuitionBase=OpenLibrary("intuition.library",0);
470 b9 GfxBase=OpenLibrary("graphics.library",0);
471 wB argc--;
472 su argv++;
473 f4 if (argc==2)
474 x1 Read_Files(*argv,argv[1]);
475 uJ else
476 VB6 if (!(Buffer=AllocMem(BUFFERSIZE,MEMF_PUBLIC)))
477 WJ1 printf("Out of memory...\\n");
478 cW6 exit(0);
479 V9B MakeAnim(GetScreen());
480 BZ FreeMem(Buffer,BUFFERSIZE);
481 ly6 CloseLibrary(IntuitionBase);
482 zN CloseLibrary(GfxBase);
483 4d1 (C) 1988 M&T
484 Ek

```

Listing 1. »anim.c« (Schluß)

Programmname: iff.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Compiler: Aztec-C V3.6

Aufrufe: cc +l -s +idmp iff.c

Programmname: iff.c

```

1 jZ0 #define MAKE_ID(a,b,c,d) ( (long)(a)<<24 | (long)(b)<
<16 | (c)<<8 | (d) )
2 Xy #define FORM MAKE_ID('F', 'O', 'R', 'M')
3 x7 #define ILBM MAKE_ID('I', 'L', 'B', 'M')
4 gT #define BMHD MAKE_ID('B', 'M', 'H', 'D')
5 WV #define CMAP MAKE_ID('C', 'M', 'A', 'P')
6 lG #define BODY MAKE_ID('B', 'O', 'D', 'Y')
7 Mm #define GRAB MAKE_ID('G', 'R', 'A', 'B')
8 cJ #define DEST MAKE_ID('D', 'E', 'S', 'T')
9 lW #define SPRT MAKE_ID('S', 'P', 'R', 'T')
10 oT #define CAMG MAKE_ID('C', 'A', 'M', 'G')
11 fX #define CRNG MAKE_ID('C', 'R', 'N', 'G')
12 7G #define CCRT MAKE_ID('C', 'C', 'R', 'T')

```



```

13 qY #define ANHD MAKE_ID('A', 'N', 'H', 'D')
14 yO #define ANIM MAKE_ID('A', 'N', 'I', 'M')
15 oO #define DLTA MAKE_ID('D', 'L', 'T', 'A')
16 ux #define DIRECT 0
17 SS #define XOR 1
18 D2 #define LONDELTA 2
19 N7 #define SHORDELTA 3
20 mN struct AnimHeader
21 I19     UBYTE operation;
22 uu     UBYTE mask;
23 hh     UWORD w,h;
24 Oj     WORD  x,y;
25 hF     ULONG abstime;
26 Av     ULONG reltime;
27 8x     UBYTE interleave;
28 2Q     UBYTE pad[21];
29 m18    };
30 toO struct AnimHeader ThisAnim = { SHORDELTA,0,0,0,0,0,1,1,1
};
31 gn union bytes4
32 jw8     char b4_name[4];
33 mX     long b4_type;
34 rn     };
35 WXO struct iff_chunk
36 bh8     union bytes4 iff_type;
37 6Q     unsigned long iff_length;
38 vr     };
39 ZaO struct form_chunk
40 nn8     union bytes4 fc_type;
41 rw     long fc_length;
42 Lu     union bytes4 fc_subtype;
43 Ow     };
44 7gO struct BitMapHeader
45 Tq8     UWORD w, h;
46 BT     UWORD x, y;
47 FE     UBYTE nPlanes;
48 Lv     UBYTE masking;
49 3d     UBYTE compression;
50 gE     UBYTE pad1;
51 kY     UWORD transparentColor;
52 Fu     UBYTE xAspect, yAspect;
53 m8     WORD  pageWidth, pageHeight;
54 B7     };
55 HoO struct ILBM_info
56 HZ8     struct BitMapHeader header;
57 pi     UBYTE cmap[32*3];
58 cF     struct BitMap bitmap;
59 GC     };
60 pMO extern FILE *MyFile;
61 hP     extern unsigned long offset,planes[8];
62 zx     #define DUMP 0
63 tL     #define RUN 1
64 gG     #define MINRUN 3
65 bV     #define MAXRUN 128
66 8a     #define MAXDAT 128
67 HK     static int pack_row(file, source, size)
68 EU1     FILE *file;
69 28     char *source;
70 3f     int size;
71 A5     char  c,
72 kuH         lastc = '\0';
73 7W1     short mode = DUMP;
74 AV     short nbuf = 0;
75 vo     short rstart = 0;
76 jF     short putsz;
77 ZK     char  buf[MAXDAT*3/2];
78 Xu     putsz = 0;
79 iE     buf[0] = lastc = *source++;
80 LL     nbuf = 1;
81 sL     size--;
82 aM     for (; size; --size)
83 Le6         buf[nbuf++] = c = *source++;
84 lL     switch (mode)
85 cSA         case DUMP:
86 SqM             if (nbuf > MAXDAT)
87 RtU                 if (file != NULL)
88 8lc                     if (putc(nbuf-2, fil
e) == EOF)
89 upk                         return(0);
90 GNe                     if (write(buf, nbuf-
1, 1, file) != 1)
91 wrG                         return(0);
92 pxU                         putsz += nbuf;

```

```

93 loQ         buf[0] = c;
94 ZZ         nbuf = 1;
95 nn         rstart = 0;
96 Tc         break;
97 sGM         if (c == lastc)
98 kdQ             if (nbuf - rstart >= MINRUN)
99 DAa                 if (rstart > 0)
100 e6f                     if (file != NULL)
101 gKk                         if (putc(rst
art-1, file)
== EOF)
return(
0);
102 72p                     if (fwrite(b
uf, rstart, 1
, file) != 1)
return(
0);
103 7mk                         putsz += rs
tart+1;
mode = RUN;
else if (rstart == 0)
mode = RUN;
else
rstart = nbuf - 1;
break;
case RUN:
if ((c != lastc) || (nbuf - rstart >
MAXRUN))
if (file != NULL)
if (putc( -(nbuf - rsta
rt - 2), file) == EOF)
return(0);
if (putc( lastc, file)
== EOF)
return(0);
putsz += 2;
buf[0] = c;
nbuf = 1;
rstart = 0;
mode = DUMP;
break;
lastc = c;
switch (mode)
case DUMP:
if (file != NULL)
if (putc(nbuf-1, file) == EOF)
return(0);
if (fwrite(buf, nbuf, 1, file) !=
1)
return(0);
putsz += nbuf+1;
break;
case RUN:
if (file != NULL)
if (putc( -(nbuf - rstart - 1), f
ile) == EOF)
return(0);
if (putc( lastc, file) == EOF)
return(0);
putsz += 2;
break;
return(putsz);
143 sJ1 static unsigned long pack_bitmap(file, bm)
144 vvO     FILE *file;
145 Tj1     struct BitMap *bm;
146 yr     short i, j;
147 iQO     short row_length;
148 sP     unsigned short plane_offset=0;
149 3r     unsigned long compressed_length=0;
150 6j     for (i=0; i<bm->Rows; i++)
151 Lp         for (j = 0; j < bm->Depth; j++)
152 AL8             if ( (row_length = pack_row(file, bm->Pla
nes[j] + plane_offset,
bm->BytesPerRow)) == 0)
153 n2G                 return(0);
154 3CO                 compressed_length += row_length;
155 yt                 plane_offset += bm->BytesPerRow;
156 l7G                 if ( compressed_length & 1)
157 UG8
158 OoO

```

Listing 2. »iff.c« enthält einige Funktionen zur Erstellung von IFF-Dateien


```

159 b38         if (file != NULL)
160 KCG             if (putc( 0, file) == EOF)
161 4z0                 return(0);
162 Qh8             compressed_length++;
163 u20 return(compressed_length);
164 YS unsigned long write_ilbm(file, colors, bits, ViewMode, com
pressed)
165 n31 FILE *file;
166 ws unsigned char *colors;
167 q7 register struct BitMap *bits;
168 vD unsigned long ViewMode;
169 dx BOOL compressed;
170 aJ struct form_chunk chunk;
171 KE struct iff_chunk ichunk;
172 9R struct BitMapHeader header;
173 Xn unsigned long bits_size;
174 H4 short i,width;
175 aA register short j;
176 Hs register short row_offset;
177 zu unsigned short MaxCol= 1 << bits->Depth;
178 S9 chunk.fc_type.b4_type = FORM;
179 ul chunk.fc_subtype.b4_type = ILBM;
180 ia if (ViewMode & HAM) MaxCol=16;
181 kX if (ViewMode & EXTRA_HALFBRITE) MaxCol=32;
182 EV chunk.fc_length=4*3*sizeof(struct iff_chunk)+MaxCol*3+siz
eof(struct BitMapHeader);
183 39 if (compressed)
184 S15     if ( (bits_size = pack_bitmap(NULL, bits)) == 0) retu
rn(0);
185 RX1 else bits_size = bits->BytesPerRow*bits->Rows*bits->D
ePTH;
186 cW chunk.fc_length += bits_size;
187 Yd chunk.fc_length += 12;
188 Fk0 /* if ((ViewMode & HAM) || (ViewMode & EXTRA_HALFBRITE)) c
hunk.fc_length+=12; */
189 tn1 if (fwrite(&chunk, sizeof(chunk), 1, file) != 1) return(0
);
190 B1 ichunk.iff_type.b4_type = BMHD;
191 lb ichunk.iff_length = sizeof(header);
192 zZ if (fwrite(&ichunk, sizeof(ichunk), 1, file) != 1) return
(0);
193 Yx header.masking = 0;
194 Iu header.pad1 = 0;
195 Sw header.transparentColor = 0;
196 Dj if (compressed) header.compression = 1;
197 kg else header.compression = 0;
198 tK width = bits->BytesPerRow*8;
199 82 header.w = width;
200 ba header.h = bits->Rows;
201 MS header.nPlanes = bits->Depth;
202 bM header.x = 0;
203 gS header.y = 0;
204 dP header.pageWidth = width;
205 SK header.pageHeight = bits->Rows;
206 zL header.xAspect = 5 * width/320;
207 90 header.yAspect = 11;
208 lv if (fwrite(&header, sizeof(header), 1, file) != 1) return
(0);
209 86 ichunk.iff_type.b4_type = CMAP;
210 Cz ichunk.iff_length = MaxCol*3;
211 Is if (fwrite(&ichunk, sizeof(ichunk), 1, file) != 1) return
(0);
212 kd if (fwrite(colors, 3*MaxCol, 1, file) != 1) return(0);
213 Ju ViewMode &= (VP_HIDE * 0xffff);
214 X5 ichunk.iff_type.b4_type = CAMG;
215 9u ichunk.iff_length = 4;
216 Nx if (fwrite(&ichunk, sizeof(ichunk), 1, file) != 1) return
(0);
217 P3 if (fwrite(&ViewMode, sizeof(ViewMode), 1, file) != 1) re
turn(0);
218 sp0 /* if (ViewMode & EXTRA_HALFBRITE)
219 cA6     ichunk.iff_type.b4_type = CAMG;
220 Ez     ichunk.iff_length = 4;
221 S2     if (fwrite(&ichunk, sizeof(ichunk), 1, file) != 1) r
eturn(0);
222 U8     if (fwrite(&ViewMode, sizeof(ViewMode), 1, file) !=
1) return(0);
223 SM5     */
224 Z11 ichunk.iff_type.b4_type = BODY;
225 Qd ichunk.iff_length = bits_size;
226 X7 if (fwrite(&ichunk, sizeof(ichunk), 1, file) != 1) return
(0);
227 lr if (compressed)

```

```

228 vT9         if (pack_bitmap(file, bits) == 0) return(0);
229 WJ1     else
230 P19         i = bits->Rows;
231 XJ         row_offset = 0;
232 3T         while (--i >= 0)
233 ZhH             for (j=0; j<bits->Depth; j++)
234 x2P                 if (fwrite( bits->Planes[j]+row_
offset, bits->BytesPerRow,
1, file) != 1)
return(0);
row_offset += bits->BytesPerRow;
235 r3f     return(chunk.fc_length+8);
236 HCX unsigned long diff_bitmap(file,bo,bn)
237 EpH
238 Nr1 FILE *file;
239 oA0 struct BitMap *bo,*bn;
240 OG1 char p=bo->Depth,j=0;
241 ev unsigned short planesize=(bo->BytesPerRow*bo->Rows)>>
1;
242 Pk
243 zR unsigned short *w=0,last_offset,new_offset,start_offset,i
;
244 Mo register unsigned short *s,*d,plane_offset;
245 ew unsigned long chunk_offset=0;
246 Lh if (file) fwrite(planes, sizeof(planes), 1, file);
247 FQ chunk_offset+=sizeof(planes);
248 EP while (p--)
249 yQ     planes[j]=chunk_offset;
250 FL9     s=bo->Planes[j];
251 nh     d=bn->Planes[j++];
252 ac     plane_offset=0;
253 cd     last_offset=0;
254 4u     i=0;
255 5Q     if (*s != *d)
256 lm         w=d;
257 vKE         if (file)
258 X7             new_offset=0;
259 WfJ             fwrite(&new_offset,2,1,file);
260 hb             fwrite(w,2,1,file);
261 AS             chunk_offset+=4;s++;d++;
262 wGE             plane_offset++;
263 Dr             while (plane_offset < planesize)
264 Y19                 if (*s == *d)
265 lBH                     switch (i)
266 kaM                         case 0:
267 JZ0                             s++;d++;
268 dA                             break;
269 GP                             case 1:
270 Of                                 if (file)
271 kK                                     new_offset=start_offset-last_
offset;
272 lQT                                     fwrite(&new_offset,2,1,file);
273 uo                                     fwrite(w,2,1,file);
274 Nf                                     last_offset=plane_offset-1;
275 Ss0                                     chunk_offset+=4;i=0;s++;d++;
276 w7                                     break;
277 OX                                     default:
278 C6                                     if (file)
279 sS                                         new_offset=last_offset-start_
offset-1;
280 vDT                                         fwrite(&new_offset,2,1,file);
281 2w                                         fwrite(&i,2,1,file);
282 Cu                                         fwrite(w,2,1,file);
283 EQ                                         last_offset=plane_offset-1;
284 b10                                         chunk_offset += 4;chunk_offset +=
(i*2);i=0;s++;d++;
285 Wh                                         break;
286 Xg     else
287 SFH         if (!i++)
288 SVM             w=d;
289 RqR             start_offset=plane_offset;
290 zD             s++;d++;
291 OXM             plane_offset++;
292 gKH         if (file)
293 6g9             putc(0xff,file);
294 8NE             putc(0xff,file);
295 90             chunk_offset+=2;
296 S49     return chunk_offset;
297 Ke1 unsigned long write_delta(file,bo,bn)
298 E70 FILE *file;
299 xD1 struct BitMap *bo,*bn;
300 bs struct form_chunk chunk;
301 hQ struct iff_chunk ichunk;
302 RL unsigned long bits_size;
303 dt chunk.fc_type.b4_type = FORM;
304 UB

```



```

305 wn chunk.fc_subtype.b4_type = ILEM;
306 qc chunk.fc_length=4+2*sizeof(struct iff_chunk)+sizeof(struct
    AnimHeader);
307 hR bits_size=diff_bitmap(NULL,bo,bn);
308 aU chunk.fc_length += bits_size;
309 lN fwrite(&chunk, sizeof(chunk), 1, file);
310 8y ichunk.iff_type.b4_type = ANHD;
311 O1 ichunk.iff_length = sizeof(struct AnimHeader);
312 Bh fwrite(&ichunk, sizeof(ichunk), 1, file);
313 oo fwrite(&ThisAnim, sizeof(struct AnimHeader), 1, file);
314 t8 ichunk.iff_type.b4_type = DLTA;
315 s5 ichunk.iff_length = bits_size;
316 F1 fwrite(&ichunk, sizeof(ichunk), 1, file);
317 vI bits_size=diff_bitmap(file,bo,bn);
318 f9 return(chunk.fc_length+8);
319 ed0 BOOL OpenAnim(FileName)
320 4o1 char *FileName;
321 af struct form_chunk AnimChunk;
322 wT if (!fopen(FileName,"w")) return FALSE;
323 mJ AnimChunk.fc_type.b4_type = FORM;
324 2V AnimChunk.fc_subtype.b4_type = ANIM;
325 RE AnimChunk.fc_length=0;
326 LV if (fwrite(&AnimChunk, sizeof(AnimChunk), 1, MyFile) != 1
    ) return FALSE;
327 gg offset=4;
328 m80 CloseAnim(FileName)
329 Dx1 char *FileName;
330 k0 ULONG Dummy;
331 LZ fclose(MyFile);
332 i1 MyFile=Open(FileName,MODE_OLDFILE);
333 OV Read(MyFile,&Dummy,4);
334 6V Write(MyFile,&offset,4);
335 Xb Close(MyFile);
336 i2 MyFile=0;
(C) 1988 M&T

```

Listing 2. »iff.c« (Schluß)

Programmname: copper.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Compiler: Aztec-C V3.6

Aufrufe: cc -I -s +idmp copper.c

Programmname: copper.c

```

1 hS0 /*****
2 rq /* Copper and Display Utilities */
3 uJ /* written by: Christian Wolf */
4 GT /* Matthias-Claudius-Weg 58 */
5 mP /* 2190 Cuxhaven 1 */
6 QI /* 04721/24269 */
7 nY /*****
8 5o #define COP1L 0x80
9 H5 #define COP2L 0x84
10 A7 #define COPJMP1 0x88
11 b9 #define COPJMP2 0x8a
12 o2 #define DIWSTRT 0x8e
13 a6 #define DIWSTOP 0x90
14 Ge #define DDFSTRT 0x92
15 iL #define DDFSTOP 0x94
16 iS #define BPL1P 0xe0
17 Xs #define BPLCON0 0x100
18 k3 #define BPLCON1 0x102
19 xE #define BPLCON2 0x104
20 zr #define BPL1MOD 0x108
21 2a #define BPL2MOD 0x10a
22 8s #define SPROP 0x120
23 EX #define COLOR00 0x180
24 hc #define INTREQ 0x9c
25 mC #define CopperEnd *((ULONG *)CurrentInstr)=0xffffffff
26 f6 UWORD *CopperList,*CurrentInstr,*LastCopperList,*DummyData
    ,*CopperList2;
27 Cs UWORD CurrentOffset=0;

```

```

28 6z char Unused[] = { "Unused" };
29 Vg char *RegisterName[] =
30 N11 { "BLTDDAT", "DMACONR", "VPOSR", "VHPOSR", "DSKDATR", "JOYODAT
    ", "JOY1DAT",
31 5I3 "CLXDAT", "ADKCONR", "POTODAT", "POT1DAT", "POTGOR", "SERDAT
    R", "DSKBYTR",
32 39 "INTENAR", "INTREQR", "DSKPTH", "DSKPTL", "DSKPTH", "DSKLEN"
    , "DSKDAT", "REFPTR",
33 oO "VPOSW", "VHPOSW", "COPCON", "SERDAT", "SERPER", "POTGO", "JO
    YTEST", "STREQU",
34 mR "STRVBL", "STRHOR", "STRLONG", "BLTCON0", "BLTCON1", "BLTAFW
    M", "BLTALWM",
35 r3 "BLTCPTH", "BLTCPTL", "BLTBPTH", "BLTBPTL", "BLTAPTH", "BLTA
    PTL", "BLTSIZE",
36 8e Unused,Unused,Unused, "BLTCON0", "BLTCON1", "BLTCON2", "BLT
    DMOD",Unused,
37 n9 Unused,Unused,Unused,Unused, "BLTCDAT", "BLTBDAT", "BLTADA
    T",Unused,
38 gx Unused,Unused,Unused, "DISKSYNC", "COP1LCH", "COP1LCL", "CO
    P2LCH", "COP2LCL",
39 Mu "COPJMP1", "COP2JMP", "COPINS", "DIWSTRT", "DIWSTOP", "DDFST
    RT", "DDFSTOP",
40 LV "DMACON", "CLXCON", "INTENA", "INTREQ", "ADKCON", "AUDOLCH",
    "AUDOLCL",
41 9J "AUDOLEN", "AUDOPER", "AUDOVOL", "AUDODAT",Unused,Unused, "
    AUD1LCH",
42 Wr "AUD1LCL", "AUD1LEN", "AUD1PER", "AUD1VOL", "AUD1DAT",Unuse
    d,Unused,
43 S7 "AUD2LCL", "AUD2LCH", "AUD2LEN", "AUD2PER", "AUD2VOL", "AUD2
    DAT",Unused,
44 tG Unused, "AUD3LCL", "AUD3LCH", "AUD3LEN", "AUD3PER", "AUD3VOL
    ", "AUD3DAT",
45 kv Unused,Unused, "BPL1PTH", "BPL1PTL", "BPL2PTH", "BPL2PTL", "
    BPL3PTH",
46 42 "BPL3PTL", "BPL4PTH", "BPL4PTL", "BPL5PTH", "BPL5PTL", "BPL6
    PTH", "BPL6PTL",
47 Qx Unused,Unused,Unused,Unused, "BPLCON0", "BPLCON1", "BPLCON
    2",Unused,
48 Su "BPL1MOD", "BPL2MOD",Unused,Unused, "BPL1DAT", "BPL2DAT", "
    BPL3DAT", "BPL4DAT",
49 x1 "BPL5DAT", "BPL6DAT",Unused,Unused,
50 hk "SPROP", "SPROP", "SPR1PTH", "SPR1PTL", "SPR2PTH", "SPR2
    PTL", "SPR3PTH",
51 Ir "SPR3PTL", "SPR4PTH", "SPR4PTL", "SPR5PTH", "SPR5PTL", "SPR6
    PTH", "SPR6PTL",
52 R8 "SPR7PTH", "SPR7PTL", "SPROPOS", "SPROCTL", "SPRODATA", "SPR
    ODATB", "SPR1POS",
53 A1 "SPR1CTL", "SPR1DATA", "SPR1DATB", "SPR2POS", "SPR2CTL", "SP
    R2DATA",
54 AP "SPR2DATB", "SPR3POS", "SPR3CTL", "SPR3DATA", "SPR3DATB", "S
    PR4POS", "SPR4CTL",
55 jG "SPR4DATA", "SPR4DATB", "SPR5POS", "SPR5CTL", "SPR5DATA", "S
    PR5DATB", "SPR6POS",
56 8A "SPR6CTL", "SPR6DATA", "SPR6DATB", "SPR7POS", "SPR7CTL", "SP
    R7DATA", "SPR7DATB",
57 gJ "COLOR00", "COLOR01", "COLOR02", "COLOR03", "COLOR04", "COLO
    R05", "COLOR06",
58 TJ "COLOR07", "COLOR08", "COLOR09", "COLOR10", "COLOR11", "COLO
    R12", "COLOR13",
59 GJ "COLOR14", "COLOR15", "COLOR16", "COLOR17", "COLOR18", "COLO
    R19", "COLOR20",
60 rp "COLOR21", "COLOR22", "COLOR23", "COLOR24", "COLOR25", "COLO
    R26", "COLOR27",
61 Iu "COLOR28", "COLOR29", "COLOR30", "COLOR31" };
62 UPO ShowCopperList(StartPointer)
63 iJ1 ULONG *StartPointer;
64 Dz ULONG Instruction;
65 D3 UWORD DataWord,Register,x,y;
66 5S do
67 5c3 ' Instruction = *StartPointer++;
68 UX if (Instruction & 0x00100000)
69 uV8 y = ( (Instruction >> 24) & 0xff);
70 wV x = ( (Instruction >> 16) & 0xfe);
71 WZ if (Instruction & 0x00000001)
72 OZD printf("Skip %x,%x\n",x,y);
73 On8 else
74 IHD printf("WAIT %x,%x\n",x,y);
75 2p3 else
76 4F8 DataWord = Instruction & 0xffff;

```

Listing 3. »copper.c« ist ein Modul, das die Programmierung der Customchips erleichtert. Die Routine wird von »play« benötigt.


```

77 HB      Register = (Instruction >> 17) & 0xff;
78 o7      printf("MOVE %x,%s\n",DataWord,RegisterName[Register]);
79 aQ2     } while ( Instruction != 0xffffffe );
80 230     #define DiwStop(h,v) DiwStrt(h,v)
81 Pw      UWORD DiwStrt(h,v) UWORD h,v; { return ((h & 0xff) | (v <
< 8)); }
82 Y2      UWORD DDFStrt(hstart,HiRes)
83 cN1      UWORD hstart;
84 pQ      BOOL HiRes;
85 v1      if (HiRes) return (((hstart-9)>>1) & 0xfffc);
86 pQ      else return (((hstart-17)>>1) & 0xffff8);
87 8m0     UWORD DDFStop(ddfstrt,w) UWORD ddfstrt,w; { return (ddfstr
t + (w>>1) - 8); }
88 Jp      CopperWait(x,y,xmask,ymask)
89 6m1     UWORD x,y,xmask,ymask;
90 Em      if (y<256)
91 L16     *CurrentInstr++ = ((y << 8) | (x >> 1)) | 0x0001
;
92 2D     *CurrentInstr++ = ((ymask << 8) | (xmask >> 1))
& 0xfffe;
93 RQ      CurrentOffset+=4;
94 L81     else
95 9J6     CopperWait(448,0xff,0x1ff,0xff);
96 7J      CopperWait(x,y & 0xff,0x1ff,0xff);
97 eQ0     CopperMove(DataWord,Register)
98 Oh1     UWORD DataWord,Register;
99 8w      *CurrentInstr++ = Register & 0x01fe;
100 JM      *CurrentInstr++ = DataWord;
101 ZY      CurrentOffset+=4;
102 8b0     CopperMoveL(Address,Register)
103 CP1      ULONG Address;
104 gP      UWORD Register;
105 e1      CopperMove((UWORD)(Address>>16),Register);
106 GN      CopperMove((UWORD)(Address & 0xffff),Register+2);
107 gY0     CopperSkip(x,y,xmask,ymask)
108 P51     UWORD x,y,xmask,ymask;
109 kK      if (y < 256)
110 e46     *CurrentInstr++ = ((y << 8) | (x >> 1)) | 0x0001
;
111 X0      *CurrentInstr++ = ((ymask << 8) | (xmask >> 1))
| 0x0001;
112 IU      CurrentInstr+=4;
113 ah0     UWORD MakeViewMode(ViewMode,Depth)
114 jp1     UWORD ViewMode,Depth;
115 Df      UWORD BplCon0=0;
116 NT      BplCon0=Depth<12;
117 WZ      if (ViewMode & HIRES) BplCon0 = HIRES;
118 nF      if (ViewMode & HAM) BplCon0 = HAM;
119 mr      if (ViewMode & DUALPF) BplCon0 = DUALPF;
120 GG      if (ViewMode & EXTRA_HALFBRITE) BplCon0 &= (HAM|DUALPF)*0
xffff;
121 Ck      if (ViewMode & LACE) BplCon0 = LACE;
122 OK      return BplCon0 | (1<<9);
123 Iu0     BOOL MakeCopperList(NewS1,NewS2,ColorMap,SecondCopperList,
LOFCopperList)
124 bu1     register struct NewScreen *NewS1,*NewS2;
125 Rp      char *ColorMap;
126 h2      UWORD *SecondCopperList;
127 Bs      UWORD *LOFCopperList;
128 9u      UWORD d,j,x,y,w=NewS1->Width,h=NewS1->Height;
129 tI      UWORD r,g,b,swap;
130 Fx      struct BitMap *bm=NewS1->CustomBitMap;
131 AM      UWORD LOFModulo=0;
132 uV      UWORD OddRows=0;
133 MC      BOOL Mode=FALSE;
134 O4      unsigned char *MyPtr=ColorMap;
135 mV      if (SecondCopperList)
136 ps6     LastCopperList=CurrentInstr=CopperList2=SecondCopper
List;
137 YY1     else if (LOFCopperList)
138 Qt6     CurrentInstr=LOFCopperList;
139 ID      OddRows=bm->BytesPerRow;
140 5s1     else
141 VU6     if (!CopperList=AllocMem(4096,MEMF_CHIP|MEMF_CLEAR)
) return FALSE;
142 TS      LastCopperList=CurrentInstr=CopperList;
143 vm1     if (ColorMap)
144 cG6     d=1<<NewS1->Depth;
145 JD      if (NewS1->ViewModes & HAM) d=16;
146 M1      j=0;

```

```

147 Yo      while (d--)
148 4eC      r=MyPtr++>>4;
149 J8      g=MyPtr++>>4;
150 au      b=MyPtr++>>4;
151 NU      CopperMove( (r << 8) | (g << 4) | b,COLORO
0 + j);
152 Jh      j+=2;
153 Tr1     if (h > 256) NewS1->ViewModes |= LACE;
154 7i      if (NewS1->ViewModes & HIRES)
155 eL6      Mode=TRUE;
156 3G      w >= 1;
157 nu1     if (NewS1->ViewModes & LACE)
158 tr6      h >= 1;
159 ag      LOFModulo=bm->BytesPerRow;
160 2S1     CopperMove(0,BPLCON1); /* kein Scrolling */
161 S1      CopperMove(0,BPLCON2); /* Priorität egal */
162 e5      CopperMove(MakeViewMode(NewS1->ViewModes,NewS1->Depth),
BPLCON0);
163 We      CopperMove(LOFModulo,BPL1MOD); /* Bitplanegröße==Displayg
röße -> Modulo=0 */
164 A6      CopperMove(LOFModulo,BPL2MOD);
165 Dy      y=41;
166 JO      x=129-(w-320);
167 H2      CopperMove(DiwStrt(x,y),DIWSTRT);
168 cw      CopperMove(DiwStop(x+w,y+h),DIWSTOP);
169 Tz      CopperMove(d=DDFStrt(x,Mode),DDFSTRT);
170 J3      CopperMove(DDFStop(d,w,Mode),DDFSTOP);
171 S1      if (bm)
172 Tv6      d=NewS1->Depth;
173 n9      j=0;
174 LZ      while (d--) CopperMoveL(bm->Planes[j++]+OddRows,BPL
1P+(j<2));
175 Ss0     /* Alternative CopperList erstellen */
176 CA      /* für interlaced Display */
177 Gv1     if ((NewS1->ViewModes & LACE) && !LOFCopperList)
178 JK6     CopperMoveL(CurrentInstr+6,COP1L);
179 hc      CopperEnd;
180 u4      if (!MakeCopperList(NewS1,0,ColorMap,0,CurrentInstr
+2)) return FALSE;
181 861     else if (NewS1->ViewModes & LACE && LOFCopperList)
182 hn6     CopperMoveL(LastCopperList,COP1L);
183 lG      CopperEnd;
184 bt1     else CopperEnd;
185 c20     /* Alternative CopperList erstellen */
186 BE      /* für double buffered Display */
187 uQ1     if (NewS2)
188 xd6     if (!MakeCopperList(NewS2,0,ColorMap,CurrentInstr+2,
0)) return FALSE;
189 mA1     return TRUE;
190 tg0     FreeCopperList();
191 yO1     if (CopperList) FreeMem(CopperList,4096);
(C) 1988 M&T

```

Listing 3. »copper.c« (Schluß)

Programmname:	play.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec-C V3.6
Aufrufe:	cc +l -s +idmp play.c

Programmname: play.c

```

1 JZ0     #define MAKE_ID(a,b,c,d) ( (long)(a)<<24 | (long)(b)<
<16 | (c)<<8 | (d) )
2 IB      #define NOTITLE NULL
3 Yz      #define FORM MAKE_ID('F', 'O', 'R', 'M')
4 y8      #define ILBM MAKE_ID('I', 'L', 'B', 'M')
5 hU      #define BMHD MAKE_ID('B', 'M', 'H', 'D')
6 XW      #define CMAP MAKE_ID('C', 'M', 'A', 'P')

```



```

7 2H #define BODY MAKE_ID('B', 'O', 'D', 'Y')
8 Nn #define GRAB MAKE_ID('G', 'R', 'A', 'B')
9 dk #define DEST MAKE_ID('D', 'E', 'S', 'T')
10 VX #define SPRT MAKE_ID('S', 'P', 'R', 'T')
11 pU #define CAMG MAKE_ID('C', 'A', 'M', 'G')
12 gY #define CRNG MAKE_ID('C', 'R', 'N', 'G')
13 8H #define CCRT MAKE_ID('C', 'C', 'R', 'T')
14 rZ #define ANHD MAKE_ID('A', 'N', 'H', 'D')
15 zI #define ANIM MAKE_ID('A', 'N', 'I', 'M')
16 pI #define DLTA MAKE_ID('D', 'L', 'T', 'A')
17 QR struct FormChunk { unsigned long Form_ID;
18 sVJ unsigned long Length;
19 s3 unsigned long ID;
20 dZH };
21 UjO struct IFFChunk { unsigned long Chunk_ID;
22 wZI unsigned long Length;
23 gcG };
24 250 #define DIRECT 0
25 aa #define XOR 1
26 LA #define LONDELTA 2
27 VF #define SHORDELTA 3
28 Sg #define VERDELTA 5
29 vW struct AnimHeader
30 Rr9 UBYTE operation;
31 33 UBYTE mask;
32 qq UWORD w,h;
33 Xs WORD x,y;
34 qO ULONG abstime;
35 J4 ULONG reltime;
36 H6 UBYTE interleave;
37 Nz UBYTE pad0;
38 Ym ULONG bits;
39 W0 UBYTE pad[18];
40 xt8 };
41 d0 struct BitMapHeader
42 Qn9 UWORD w, h;
43 8Q UWORD x, y;
44 WC UBYTE planes;
45 Is UBYTE masking;
46 0a UBYTE compression;
47 dB UBYTE pad1;
48 hV UWORD transparentColor;
49 Cr UBYTE xAspect, yAspect;
50 j5 WORD pageWidth, pageHeight;
51 848 };
52 VZ0 #define NextLong ReadLongWord(MyFile)
53 eC #define LOOP 2
54 1S #define CONT 1
55 kO #define ONCE 0
56 0I struct BitMapHeader BitMapHeader;
57 69 struct AnimHeader AnimHeader;
58 Ps struct IntuitionBase *IntuitionBase;
59 jM struct GfxBase *GfxBase;
60 9D struct BitMap bml,bm2;
61 10 extern UWORD *CopperList,*CopperList2;
62 fK struct NewScreen NewS1 = {
63 9a7 0,0,640,256,3,0,1,HIRES,CUSTOMSCREEN,NULL,NOTITLE,0
, &bml };
64 kq0 struct NewScreen NewS2 = {
65 Df7 0,0,640,256,3,0,1,HIRES,CUSTOMSCREEN,NULL,NOTITLE,0
, &bml };
66 dK0 char ColorMap[192];
67 3g struct DLTAChunk { struct DLTAChunk *Next;
68 NzJ unsigned short Size;
69 AfH } *Start=0,*Current,*Last=0;
70 GAO char *BodyChunk;
71 Jq unsigned long BodySize;
72 pI BOOL Mode=ONCE,FirstTime=TRUE;
73 qF UBYTE DeltaMode;
74 8h ULONG MuluTable[626];
75 x9 InitMuluTable(bpr)
76 oQ1 short bpr;
77 Pm short x;
78 E4 for (x=0;x<626;x++) MuluTable[x]=bpr*x;
79 Ys0 unsigned long ReadLongWord(MyFile)
80 sD1 unsigned long MyFile;
81 kF unsigned long Buffer;
82 2v Read(MyFile,&Buffer,4);
83 wT return Buffer;
84 gG0 PlayAnim()
85 001 MakeCopperList(&NewS1,&NewS2,ColorMap,0,0);

```

```

86 Sp Forbid();
87 jU Disable();
88 kU if (DeltaMode==SHORDELTA) ShortDelta(&BitMapHeader);
89 CJ if (DeltaMode==VERDELTA) VertDelta(&BitMapHeader);
90 lZ Permit();
91 gZ Enable();
92 lHO ShortDelta(bmh)
93 oG1 struct BitMapHeader *bmh;
94 WH char *(*PlanePtr)[8]=&bml.Planes[0];
95 7t char **CurrentPtr=&bm2.Planes[0];
96 M2 #asm
97 bgD xdef _CopperList,_CopperList2
98 L50 FOREVER: pea _BitMapHeader
99 MUD jsr _WriteBody
100 Do add.w #4,sp
101 OS pea _bm2
102 KN pea _bml
103 nx jsr _CopyBitMap
104 T8 add.w #8,sp
105 IL movem.l a4/a5,-(a7)
106 KR move.l _Start,a0 ;a0 = Zeiger auf Date
n
107 PN move.l -4(a5),d7 ;Zeiger auf bml.Plane
s[0]
108 ce move.l -8(a5),d6 ;Zeiger auf bm2.Plane
s[0]
109 Kb move.l _CopperList,d5
110 kg move.l _CopperList2,d4
111 gR move.w _Mode,d2
112 vt move.w #$0000000110100000,$dff096 ;dma sp
erren
113 6c move.l d5,$dff080
114 az clr.w $dff088
115 BI move.w #0111111111111111,$dff09c
116 RX move.w #$ffff,d3
117 wX move.l a0,a4
118 g9 move.w #32,$dff09c ; VBL-RequestBit 15sc
hen
119 nv0 WaitNext: move.w $dff01e,d0 ; auf den nächsten VBL
warten
120 zID and.w #32,d0
121 PH beq WaitNext
122 TY move.w #$8000,$dff02a ; LOF B
it setzen
123 EU move.w #$1000000110000000,$dff096 ;Copper
und Plane DMA an
124 GT0 NextFrame: lea 6(a0),a1 ; Zeiger auf Differen
zdaten
125 MdD move.l a1,a5 ; a1 sichern
126 wz move.l d6,a3 ; Zeiger auf Planes[0
] in a3
127 wF bra dstart ; Schleife starten
128 8B0 dloop: move.l d0,a2 ; Offset in a2
129 b9D adda.l a5,a2 ; absolute Adresse be
rechnen
130 73 move.l (a3)+,a6 ; a6=Zeiger auf Bitpl
ane
131 iV0 wloop: move.w (a2)+,d1 ; Datenwort holen
132 TXD bmi next ; negativ ? ja -> ne
xt
133 Sc xt add.w d1,d1 ; d1 *= 2
134 qD add.w d1,a6 ; Adresse des Wortes
zu a6 addieren
135 FR move.w (a2)+,(a6) ; in Bitplane schreib
en
136 sM bra wloop ; nächstes Wort
137 H90 next: eor.w d3,d1
138 7gD beq dstart ;d1=0 ? ja -> next p
lane
139 S2 add.w d1,d1 ; d1 verdoppeln
140 K1 add.w d1,a6 ; Adresse des ersten W
ortes in der Bpl
141 lv move.w (a2)+,d0 ; Anzahl holen und ...
142 r6 bra copystart
143 XC0 copyloop: move.w (a2)+,(a6)+
144 nc copystart: dbra d0,copyloop ; kopieren

```

Listing 4. Mit »play.c« lernen die von »anim.c« erzeugten Bilder laufen


```

145 lnd      suba.w    #2,a6      ; BitplaneZeiger korr
146 zd      igleren
147 eio      dstart:  bra      wloop      ; weiter geht's
erhalb des Chunks      move.l    (a1)+,d0      ; Zeiger auf Daten inn
148 EKD      bne.s     dloop      ; ungleich 0 ? ja ->
nächste Plane
149 Rw      and.w     #1024,$dff016 ; rechte Maustaste g
edrückt ?
150 35      beq      Exit
151 bP      andi.b    #64,$bfe001 ; linke Maustaste gedr
ückt
152 HJ      bne      skip
153 Oc0      WaitMouse1: andi.b    #64,$bfe001
154 ADD      beq      WaitMouse1
155 JEO      skip:    exg      d6,d7      ; Zeiger auf Planes[0]
austauschen
156 RyD      exg      d4,d5      ; Zeiger auf CopperList
austauschen
157 x7      move.l    (a0),a0      ; Ende der Animation
158 pR      cmpa.l    #0,a0      ; erreicht ???
159 Xs      bne.s     NotEnd
160 YH      cmpi.w    #0,d2
161 OI      beq.s     Exit
162 dN      cmpi.w    #1,d2
163 HB      beq.s     Loop
164 K1      movem.l   (a7)+,a4/a5
165 44      move.w     #0000000110100000,$dff096 ;DMA sp
erren
166 We      bra      FOREVER
167 VCO      Loop:    move.l    a4,a0      ; wieder von Anfang an
168 8aD      move.l    (a0),a0
169 8g0      NotEnd:  move.w     #32,$dff09c
170 pV      WaitFrame: move.w     $dff01e,d0      ; auf nächsten
171 BjD      and.w     #32,d0      ; VBL
172 Qt      beq      WaitFrame      ; warten...
173 3x      move.w     #8000,$dff02a ; LOF Bit setzen (
falls Interlace)
174 It      move.l    d5,$dff080      ; neue Copperliste
175 QN      clr.w     $dff088      ; und in CopperPC l
aden
176 vn      bra.s     NextFrame      ; nächstes Bild bea
reiten
177 4Jo      Exit:    movem.l   (a7)+,a4/a5
178 HHD      move.w     #0000000110100000,$dff096 ;DMA sp
erren
179 Gv      move.l    _GfxBase,a0      ; alte CopperList
180 sd      move.l    38(a0),$dff080 ; zurück schreiben
181 RG      clr.w     $dff088      ; und starten
182 JV      move.w     #1000000110100000,$dff096 ; DMA w
ieder an
183 bV1      #endasm
184 ALO      VertDelta(bmh)
185 Ik1      struct BitMapHeader *bmh;
186 O1      char *(*PlanePtr)[8]=&bm1.Planes[0];
187 bN      char **CurrentPtr=&bm2.Planes[0];
188 FR      short BytesPerRow=((bmh->w)>>4)<<1;
189 Kp      ULONG Swap1,Swap2,Start;
190 sY      #asm
191 7CD      xdef      _CopperList,_CopperList2
192 Du      move.l    _CopperList,-14(a5)
193 Hq      move.l    _CopperList2,-18(a5)
194 PQ0      FOREVER1: pea      _BitMapHeader
195 u2D      jsr      _WriteBody
196 1M      add.w     #4,sp
197 w0      pea      _bm2
198 sv      pea      _bm1
199 LV      jsr      _CopyBitMap
200 1g      add.w     #8,sp
201 qt      movem.l   a4/a5,-(a7)
202 sz      move.l    _Start,a0      ;a0 = Zeiger auf Date
n
203 Av      move.w     _Mode,d2
204 PN      move.w     #0000000110100000,$dff096 ;dma sp
erren
205 Bh      move.l    _CopperList,$dff080
206 4T      clr.w     $dff088
207 fm      move.w     #0111111111111111,$dff09c
208 kr      move.l    a0,-22(a5)
209 4Q      move.w     -10(a5),d4      ; d4=Anzahl der Spalte
n
210 8f      lea      _MuluTable,a4

```

```

211 Be      move.w     #32,$dff09c ; VBL-RequestBit lösc
hen
212 aW0      WaitNext1: and.w     #32,$dff01e ; auf den nächsten VB
L warten
213 GPD      beq      WaitNext1
214 x2      move.w     #8000,$dff02a ; LOF B
it setzen
215 1y      move.w     #1000000110000000,$dff096 ;Copper
und Plane DMA an
216 iv0      NextFrame1: lea      6(a0),a1 ; Zeiger auf Differen
zdaten
217 5PD      move.l    a1,d5 ; a1 sichern
218 Hy      move.l    -8(a5),a3 ; Zeiger auf Planes[0
] in a3
219 Fp      bra      dstart1 ; Schleife starten
220 au0      move.l    d0,a2 ; Offset in a2
221 BmD      adda.l    d5,a2 ; absolute Adresse be
rechnen
222 tv      move.l    (a3)+,d6 ; d6=Zeiger auf Bitpl
ane
223 VT      move.w     d4,d3 ; d3=Zähler für Spalt
en
224 NL      subq.w    #1,d3
225 lK0      sloop:   clr.w     d0
226 A0D      move.b    (a2)+,d0 ; Anzahl der Opcodes
227 2x      move.l    d6,a6 ; Zeiger auf Bitplane
in a6
228 Vc      bra      OpStart ; Schleife starten
229 jh0      OpLoop:  clr.w     d1 ; d1 löschen
230 eSD      move.b    (a2)+,d1 ; Opcode holen
231 It      bne.s     not_same ; ungleich 0 ? ja ->
not_same
232 ox      move.b    (a2)+,d1 ; Opcode = same : d1=
Anzahl
233 KQ      subq.w    #1,d1
234 bF      move.b    (a2)+,d7 ; Datum holen
235 870      move.b    d7,(a6) ; und kopieren
236 5SD      adda.w    d4,a6 ; Zeiger um Spaltenan
zahl erhöhen
237 Wy      dbra      d1,floop ; nächstes Bytes
238 p3      dbra      d0,OpLoop
239 TD      addq      #1,d6
240 Gf      dbra      d3,sloop
241 Jp      move.l    (a1)+,d0 ; Zeiger auf Daten in
nerhalb des Chunks
242 N3      bne.s     dloop1 ; ungleich 0 ? ja ->
nächste Plane
243 Da      bra      RMouse
244 vJ0      not_same: bmi.s     uniq ; bit 7 gesetzt ? ja
-> uniq
245 CcD      add.w     d1,d1
246 Pb      add.w     d1,d1 ; d1 *= 4
247 BL      adda.l    (a4,d1),a6
248 zD      dbra      d0,OpLoop
249 dN      addq      #1,d6
250 Qp      dbra      d3,sloop
251 Dr      move.l    (a1)+,d0 ; Zeiger auf Daten i
nnerhalb des Chunks
252 v4      bne.s     dloop1 ; ungleich 0 ? ja -
> nächste Plane
253 Nk      bra      RMouse
254 nX0      uniq:    and.b     #127,d1 ; opcode = uniq : Bi
t 7 löschen
255 gmD      subq.w    #1,d1
256 Zt0      copyloop1: move.b    (a2)+,(a6)
257 mVD      adda.w    d4,a6 ; a6 um Spaltenanzahl
erhöhen
258 8u      dbra      d1,copyloop1
259 RG0      OpStart: dbra      d0,OpLoop
260 oYD      addq      #1,d6
261 ME0      sstart:  dbra      d3,sloop
262 NN      dstart1: move.l    (a1)+,d0 ; Zeiger auf Daten in
nerhalb des Chunks
263 iOD      bne.s     dloop1 ; ungleich 0 ? ja ->
nächste Plane
264 k00      RMouse:  and.w     #1024,$dff016 ; rechte Maustaste g
edrückt ?
265 YUD      beq      Exit1
266 XY      andi.b    #64,$bfe001 ; linke Maustaste ge
drückt
267 m8      bne      skip1
268 LV0      WaitMouse2: andi.b    #64,$bfe001

```



```

269 76D      beq      WaitMouse2
270 Fx0 skip1: move.l   -4(a5),d7
271 1PD      move.l   -8(a5),-4(a5)
272 tv       move.l   d7,-8(a5)      ; Zeiger auf Planes
[0] austauschen
273 h4       move.l   -14(a5),d7
274 kw       move.l   -18(a5),-14(a5)
275 lh       move.l   d7,-18(a5)    ; Zeiger auf CopperList
austauschen
276 s2       move.l   (a0),a0      ; Ende der Animation
277 kM       cmpa.l   #0,a0        ; erreicht ???
278 Ht       bne.s    NotEnd1
279 TC       cmpi.w   #0,d2
280 x1       beq.s     Exit1
281 YI       cmpi.w   #1,d2
282 qe       beq.s     Loop1
283 Fg       movem.l  (a7)+,a4/a5
284 zz       move.w   #0000000110100000,$dff096 ;DMA sp
erren
285 Xw       bra      FOREVER1
286 OT0 Loop1: move.l   -22(a5),a0    ; wieder von Anfang
an
287 3VD      move.l   (a0),a0
288 I70 NotEnd1: move.w   #32,$dff09c
289 DW WaitFrame1: and.w   #32,$dff01e    ; auf nächsten
290 yLD      beq      WaitFrame1    ; warten...
291 xr       move.w   #8000,$dff02a  ; LOF Bit setzen (
falls Interlace)
292 Ix       move.l   -14(a5),$dff080 ; neue Copperl
iste setzen
293 KH       clr.w     $dff088      ; und in CopperPC l
aden
294 TF       bra.s     NextFrame1   ; nächstes Bild bea
rbeiten
295 wb0 Exit1: movem.l  (a7)+,a4/a5
296 BBD      move.w   #0000000110100000,$dff096 ;DMA sp
erren
297 Ap       move.l   _GfxBase,a0   ; alte CopperList
298 mX       move.l   38(a0),$dff080 ; zurück schreiben
299 LA       clr.w     $dff088      ; und starten
300 DP       move.w   #1000000110100000,$dff096 ; DMA w
ieder an
301 VP1      #endasm
302 G00 WriteBody(bmh)
303 Ce1      struct BitMapHeader *bmh;
304 uf       char *(*PlanePtr)[8]=&bm1.Planes[0];
305 oD       char **CurrentPtr,*ReadPointer,*pointer;
306 rh       unsigned short h=bmh->h,bpl=bmh->w>>3,d=bmh->Planes,
PlaneOffset=0;
307 HN       unsigned short j,count;
308 NI       char n,x;
309 nT       #asm
310 iXD      clr.l     'd0
311 lb       clr.l     d1
312 of       clr.l     d2
313 qc       move.l   _BodyChunk,a0 ;a0 = Zeiger auf Dat
en
314 ih       move.l   -4(a5),a1      ;Zeiger auf Planes[0
]
315 I6       move.w   -18(a5),d0 ; d0 = Anzahl der Zeilen
316 bR       move.w   -20(a5),d1 ; d1 = Bytes pro Zeile
317 rf       move.w   -22(a5),d2 ; d2 = Anzahl der Planes
318 Zu       clr.l     d4           ; d4=PlaneOffset
319 om       clr.l     d6           ; d6=n=0
320 Eu       clr.l     d5           ; PlaneOffset=0
321 DO       clr.l     d3           ;
322 7X       clr.l     d7
323 Ga       bra      hstart
324 mk0 hloop: move.w   d2,d3
325 wxD      move.l   a1,a2          ;a2=CurrentPtr
326 xP       bra      pstart
327 eu0 ploop: move.l   (a2)+,a3      ;a3=pointer
328 pSD      add.l     d4,a3          ; += PlaneOffset
329 yL       clr.b     d5           ; count=0
330 cJO      cloop:   clr.w     d6
331 3AD      move.b    (a0)+,d6      ;n=ReadPointer++
332 dK       tst.b     d6
333 s1       bmi      else
334 X1       addq.b     #1,d6        ;n++
335 nk       add.b     d6,d5        ;count += n
336 XX       bra      cpstart
337 WRO      cploop:   move.b    (a0)+,(a3)+

```

```

338 az      cpstart:   dbra      d6,cploop
339 7MD      bra      cstart
340 J10      else:      move.b    (a0)+,d7 ;x=ReadPointer++
341 qPD      neg.b      d6          ;n = -n
342 fq       addq.b     #1,d6      ;n++
343 vs       add.b      d6,d5      ;count += n
344 6H       bra      xcstart
345 by0 xcloop: move.b    d7,(a3)+
346 NL      xcstart:   dbra      d6,xcloop
347 Kq       cstart:   cmp.b      d1,d5
348 QkD      bne.s      cloop
349 VJO      pstart:   dbra      d3,ploop
350 9xD      add.w      d1,d4      ;PlaneOffset+=bpl
351 Js0      hstart:   dbra      d0,hloop
352 KE1      #endasm
353 f40      /*
354 521      ReadPointer=BodyChunk;
355 AU       while (h--)
356 6I7      j=d;
357 70       CurrentPtr=PlanePtr;
358 lh       while (j--)
359 q1D      pointer=*CurrentPtr++ + PlaneOffset;
360 9s       count=0;
361 12       while (count != bpl)
362 LNG      if ((n = *ReadPointer++) >= 0)
363 nqL      n++;
364 aC       count += n;
365 rL       while (n--) *pointer++=*ReadPointer++
;
366 jWG      else
367 VRL      x = *ReadPointer++;
368 Pz       n = -n;
369 tw       n++;
370 gI       count += n;
371 4x       while (n--) *pointer++=x;
372 1n8      PlaneOffset += bpl;
373 4T1      /*
374 K10 CopyBitMap(s,d)
375 4G1      struct BitMap *s,*d;
376 x0       char p=s->Depth,j=0;
377 ua       register unsigned long c;
378 ZU       unsigned long size = s->BytesPerRow * s->Rows;
379 kp       register char *sp,*dp;
380 5X       while (p--)
381 C09      c=size;
382 JV       sp=s->Planes[j];
383 y4       dp=d->Planes[j++];
384 mp       while (c--) *dp++=*sp++;
385 RNO      FreePlanes(bm)
386 Dr1      register struct BitMap *bm;
387 Ns       unsigned short p=bm->Depth,bpr=bm->BytesPerRow,r=bm->R
ows;
388 5R       unsigned short planesize=bpr*r;
389 jh       char j=0;
390 XE       APTR PlanePtr;
391 qH       while (p--) if (PlanePtr=bm->Planes[j++]) FreeMem(PlaneP
tr,planesize);
392 Zf0      CloseAnim(bm)
393 Aq1      struct DLTACHunk *Temp;
394 U1       FreeCopperList();
395 15       FreePlanes(&bm1);
396 p8       FreePlanes(&bm2);
397 qI       if (BodyChunk) FreeMem(BodyChunk,BodySize);
398 bN       Current=Start;
399 L6       while (Current)
400 tI3      Temp=Current->Next;
401 oo       FreeMem(Current,Current->Size);
402 jI       Current=Temp;
403 7e1      Start=Current=Last=0;
404 Vr0      MakeColorMap(ThisView)
405 CL1      struct ViewPort *ThisView;
406 H9       char *c=ColorMap;
407 Ih       unsigned short i,d=1<<BitMapHeader.Planes;
408 jY       unsigned short r,g,b;
409 S4       for (i=0;i<d;i++)
410 kn7      r=*c++>>4;
411 PH       g=*c++>>4;
412 G3       b=*c++>>4;
413 h1       SetRGB4(ThisView,i,r,g,b);
414 tX0      BOOL InitRaster(bm,w,h,d)

```

Listing 4. »play.c« (Fortsetzung)


```

415 JC1 struct BitMap *bm;
416 H4 unsigned short w,h,d;
417 KW unsigned short planesize = (w>>3)*h;
418 zg APTR PlanePtr;
419 DB char j=0;
420 qI InitBitMap(bm,d,w,h);
421 yE while (d-->0)
422 k19 if (PlanePtr=AllocMem(planesize,MEMF_CHIP))
423 wOE bm->Planes[j++]=PlanePtr;
424 Kg9 else return(FALSE);
425 ay1 return TRUE;
426 pw0 BOOL InitDisplay(bmh)
427 nD1 register struct BitMapHeader *bmh;
428 3Y if (!InitRaster(&bm1,bmh->w,bmh->h,bmh->Planes)) retur
429 Ab n FALSE;
430 YG if (!InitRaster(&bm2,bmh->w,bmh->h,bmh->Planes)) retur
431 4J n FALSE;
432 b9 NewS1.ViewModes=NewS2.ViewModes=0;
433 Ne if (bmh->h > 256) NewS1.ViewModes=(NewS2.ViewModes |= L
434 SE ACE);
435 94 if (bmh->w > 352) NewS1.ViewModes=(NewS2.ViewModes |= H
436 O6 IRES);
437 bJ NewS1.ViewModes = (NewS2.ViewModes |= VP_HIDE);
438 I9 NewS1.Width=bmh->w;
439 XB NewS1.Height=bmh->h;
440 pD NewS1.Depth=bmh->Planes;
441 uP0 NewS2.Width=bmh->w;
442 9S1 NewS2.Height=bmh->h;
443 1I NewS2.Depth=bmh->Planes;
444 1p return TRUE;
445 1I unsigned long ReadILBM(MyFile,End)
446 Oo unsigned long MyFile,End;
447 n63 unsigned long ID,Length,Offset;
448 cg BOOL Error=FALSE;
449 t8 Offset=0;
450 iu while (Offset < End && Error==FALSE )
451 3m ID=NextLong;
452 nC5 Length=NextLong;
453 Gk if (Length & 1) Length++;
454 SX if (Length & 1) Length++;
455 GP Offset+=Length+8;
456 Pu switch (ID)
457 VL case BMHD:
458 JS Read(MyFile,&BitMapHeader,sizeof(BitMapHeader));
459 vH if (!InitDisplay(&BitMapHeader)) Error=TRUE;
460 Oo break;
461 MV case CMAP:
462 UC Read(MyFile,ColorMap,Length);
463 qw break;
464 ZF case CAMG:
465 2v NewS1.ViewModes=NewS2.ViewModes=NextLong ]. VP_HIDE;
466 Ra break;
467 3S case BODY:
468 IT BodySize=Length;
469 Ns if (!BodyChunk=AllocMem(Length,MEMF_PUBLIC)) Error=
470 EUA TRUE;
471 aa else Read(MyFile,BodyChunk,Length);
472 CO break;
473 ZtF case ANHD:
474 o3 Read(MyFile,&AnimHeader,Length);
475 eJA if (FirstTime)
476 bk5 FirstTime=FALSE;
477 kJ DeltaMode=AnimHeader.operation;
478 yX if (DeltaMode != SHORTDELTA && DeltaMode != VERT
479 aJA DELTA)
480 u9 printf("Unknown data copression...\n");
481 aN5 Error=TRUE;
482 XQA InitMuluTable((BitMapHeader.w>>4)<<1);
483 mo break;
484 SD case DLTA:
485 o1 if (!Current=AllocMem(Length+sizeof(struct DLTACHunk
486 cq ),MEMF_PUBLIC))
487 du printf("Out of Memory!\n");
488 nw5 Error=TRUE;
489 Bb else
490 5p if (!Start) Start=Current;
491 3C if (Last) Last->Next=Current;
492 Last=Current;
493 Read(MyFile,Current+1,Length);
494 Current->Next=0;
495 Current->Size=Length+sizeof(struct DLTACHunk);
496 break;
497 case GRAB:
498 case DEST:
499 case SPRT:

```

```

492 vb case CCRT:
493 MO case CRNG:
494 FP printf("unused found.\n");
495 Zr Seek(MyFile,Length,0);
496 v4 break;
497 Jd default:
498 ZI printf("Sorry, but this is a corrupted IFF-File\n");
499 DS Error=TRUE;
500 z8 break;
501 zm1 return Error;
502 Ze0 BOOL ReadIFF(MyFile)
503 h21 unsigned long MyFile;
504 eV unsigned long ID,Length,Offset=0,End;
505 ko BOOL Error=FALSE;
506 JG if ((ID=NextLong) == FORM)
507 7u6 End=NextLong;
508 4W if ((ID=NextLong) != ANIM) Error=TRUE;
509 AR else Offset+=4;
510 Hh while (Offset < End && Error==FALSE)
511 o18 if ((ID=NextLong) == FORM)
512 eID Length=NextLong;
513 VC if ((ID=NextLong) == ILBM)
514 kII Offset+=8;
515 AS if (ReadILBM(MyFile,Length-4)) Error=TRU
516 4C E;
517 AxD else Offset+=Length;
518 1dI else
519 Xm printf("Sorry, but this is an mangled IF
520 NP8 F-File...\n");
521 E11 Error=TRUE;
522 by6 else
523 bq printf("This is not an IFF File.\n");
524 M91 Error=TRUE;
525 Xt0 return Error;
526 uU1 main(argc,argv)
527 W4 int argc;
528 6R char **argv;
529 SA unsigned long MyFile;
530 t8 char c;
531 pr IntuitionBase=OpenLibrary("intuition.library",0);
532 hA GfxBase=OpenLibrary("graphics.library",0);
533 90 printf("\n*****\n");
534 v1 printf("\n*** Delta Anim Player 0.8 ***\n");
535 cx printf("\n*** written in Aztec C by: Christian Wolf ***\n");
536 Bw printf("\n*** Copyright (c) 1988 by CWF ***\n");
537 h6 printf("\n*****\n");
538 ga argc--;
539 Oo if (!argc) printf("Need a filename as input...\n");
540 bS argv++;
541 1A6 if ((*argv)[0] == '-')
542 e9 argc--;
543 OJF switch (c=(*argv)[1])
544 TG case 'c':
545 OV case 'C':
546 js Mode=CONT;
547 Vx break;
548 yu case 'l':
549 6h case 'L':
550 nw Mode=LOOP;
551 bV break;
552 B2 default:
553 qz printf("illegal option! -%c\n",c);
554 F36 break;
555 Ih1 argv++;
556 gb8 while(argc-->0)
557 TtD if (!MyFile=Open(*argv++,MODE_OLDFILE))
558 7Y printf("Can't open specified file...\n");
559 2j8 continue;
560 K9 printf("Loading animation into memory... Please st
561 FGD and by.\n\n");
562 Th8 if (!ReadIFF(MyFile))
563 DH PlayAnim();
564 W21 CloseAnim();
565 OX Close(MyFile);
566 OX CloseLibrary(GfxBase);
567 M&T CloseLibrary(IntuitionBase);
568 (C) 1988 M&T

```

Listing 4. »play.c« (Schluß)

Fonts mit allen Tricks

Durch die Darstellung von Texten als Grafik stehen dem Amiga-Anwender zahlreiche Zeichentypen zur Wahl. Eine Vielzahl verschiedener Standardfonts steht bereit. Aber kreative Amiga-Benutzer möchten natürlich auch eigene Zeichensätze entwerfen oder bestehende ändern. Genau dieser Wunsch wird mit dem »Fontdesigner« Wirklichkeit. Bearbeiten Sie mit dem Fontdesigner alle Amiga-Zeichensätze, die eine Höhe von 50 Punkten nicht überschreiten. Sogar ein Programmiermodus ist enthalten, mit dem gezielt Zeichensätze oder auch Teile daraus geändert werden. In Tabelle 1 (Seite 53) finden Sie eine Beschreibung aller Fähigkeiten dieses Programms.

Der Fontdesigner ist in der Programmiersprache C geschrieben. Geben Sie die Listings 1 bis 6 mit dem Checksummer (Seite 159) ein. Nachdem die Listings alle einzeln auf einer Diskette gespeichert wurden, verwenden Sie die Hinweise in Tabelle 2a (für Aztec-Compiler V3.4a) beziehungsweise Tabelle 2b (für Lattice-Compiler), um das Programm zu compilieren und zu linken. Besitzen Sie die

Programmservice-Diskette, entfällt diese Arbeit natürlich,

Der Amiga wird selbst mit vielen verschiedenen Zeichensätzen spielend fertig. Bisher gab es aber nur wenige Programme, mit denen Zeichensätze zu bearbeiten oder selbst zu erstellen sind. Mit dem »Fontdesigner« wird das anders.

die lauffähige Version befindet sich auf dieser.

Bevor Sie das Programm starten, noch einige Tips:

Der Fontdesigner benötigt den CLI-Befehl »Assign« beim Laden von Zeichensätzen.

Letzte Vorbereitungen

Falls Sie nur ein Laufwerk besitzen, sollten Sie den Fontdesigner in der Startup-Sequence starten und alle Fonts auf dieser Diskette verwalten, um sich häufiges Wechseln der Diskette zu ersparen. Außerdem sollten Sie in jedem Fall dafür sorgen, daß die Datei »Assign« im Verzeichnis »c« und die Datei »diskfont.library« im Verzeichnis »libs« auf Ihrer Bootdiskette zu finden sind. Besitzen Sie zwei Laufwerke, booten Sie einfach mit Ihrer Workbench; haben Sie nur ein Laufwerk, so sollten Sie die beiden genannten Da-

teien auf Ihre Fontdesigner-Diskette in die richtigen Verzeichnisse kopieren.

Nachdem Sie das Programm mit

Fontdesigner <RETURN>

vom CLI aus gestartet haben, gelangen Sie in den Hauptbildschirm (Bild 1). Von hier aus werden alle Funktionen des Programms kontrolliert; dies geschieht durch Anklicken des gewünschten »Schalters«. Der Hauptbildschirm ist — auch wenn er nicht so aussieht — ein ganz normales Amiga-Fenster, so daß mit Hilfe der Gadgets oben rechts andere Fenster nach vorne geklickt werden können, um das Multitasking des Amiga zu nutzen.

Den größten Teil des Bildschirms nimmt die Zeichenmatrix ein. Da sie 64 Kästchen breit und 50 Kästchen hoch ist, verarbeitet das Programm Fonts mit einer maximalen Größe von 64 mal 50 Punkten.

Um einen Punkt in der Ma-

trix zu setzen, wird er einfach mit der linken Maustaste angeklickt, die rechte Maustaste dient zum Löschen bereits gesetzter Punkte. Sie können durch Überfahren der Matrix mit niedergehaltener linker Maustaste in diese sozusagen »hineinmalen«.

Wichtig: Base- und Helpline

In der Matrix befinden sich ferner ein Rechteck und zwei Linien in oranger Farbe (alle Farbangaben gelten für eine Workbench, auf der die Bildschirmfarben der Original-Workbench-Diskette nicht verändert wurden). Das Rechteck umgrenzt das Feld, das beim Speichern eines Zeichensatzes auf die Diskette geschrieben wird — es gibt also die Größe der Zeichen an. Die waagerechte Linie ist die sogenannte Baseline, die bei der Gestaltung von Unterlängen (etwa beim »g«) wichtig ist (Bild 2). Die senkrechte Linie ist die »Helpline« — sie hat bei den Funktionen »spiegeln« und »scrollen« eine wichtige Bedeutung.

Das Feld neben dem »PROGRAMM«-Schalter dient zur Anwahl des Zeichens, das man bearbeiten will. Bevor eine Veränderung des Zeichensatzes durchgeführt werden kann, müssen natürlich erst einmal die vorhandenen Schriftarten (Fonts) von Diskette in den Arbeitsspeicher geladen werden. Verwenden Sie dazu den Schalter »LOAD«. Genaues hierzu finden Sie weiter unten.

Im obersten Feld steht der ASCII-Code des aktuellen Zeichens; klickt man dieses Feld an, kann ein neuer Code mit der Tastatur eingegeben werden. Mit den beiden Pfeilen wird die Zahl im obersten Feld hinauf- und heruntergezählt. Wenn Sie einen bestimmten Buchstaben bearbeiten wollen, tippen Sie ihn einfach per Tastatur ein.

Rechts neben der Zeichenmatrix befinden sich zwei kleine Fenster. Das obere Fenster stellt den in der Zeichenmatrix vergrößert dargestellten Buchstaben in Normalgröße dar. Im unteren Fenster ist der Inhalt eines Zwischenspeichers zu sehen, der von den »SNP«- und »CPY«-Funktionen verwendet wird.

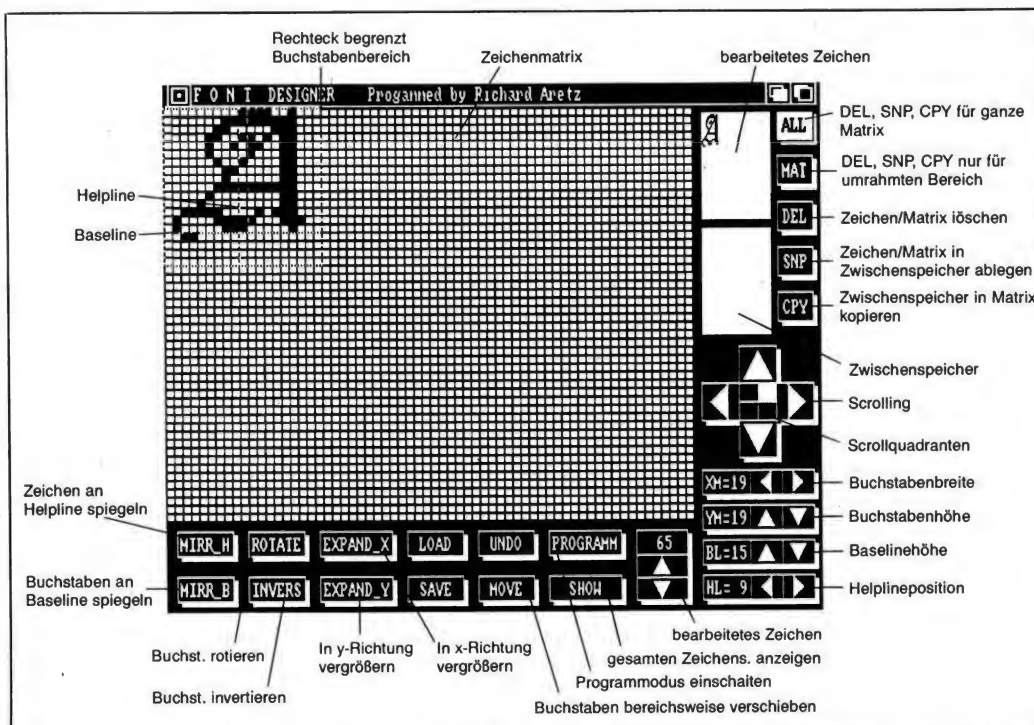


Bild 1. Der »Fontdesigner« gestattet kreative Veränderungen an den Zeichensätzen. Sogar die Programmierung diverser Änderungen ist vorgesehen.

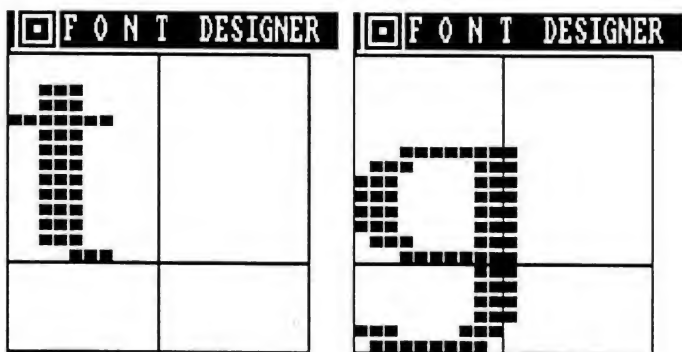


Bild 2. So muß die Baseline bei Unterlängen verwendet werden, damit das »g« richtig auf der Zeile sitzt

Oben rechts im Hauptdisplay befinden sich fünf Schalter mit den Aufschriften ALL, MAT, DEL, SNP und CPY.

ALL und MAT

Ist ALL aktiviert, gelten die unteren drei Befehle für die gesamte Zeichenmatrix; ist MAT eingeschaltet, gelten sie nur für den Bereich des gelben Rechtecks.

DEL

Der gerade bearbeitete Buchstabe wird gelöscht.

SNP

Kopiert den aktuellen Buchstaben in einen Zwischenspeicher.

CPY

Kopiert den Zwischenspeicher in die Matrix.

Unterhalb der gerade beschriebenen Schalter befindet sich das **Scrollfeld**. Es besteht aus den Richtungssymbolen und dem Quadrantenfeld in der Mitte. Die Helpline und die Baseline teilen die orange umrandete Buchstabenmatrix in vier Quadranten, die alle einzeln in alle Richtungen gescrollt werden können. Bevor sie also den Scroll-Vorgang mit einem Mausklick auf einen der Richtungspfeile starten, müssen Sie den gewünschten Quadranten im Quadrantenfeld anklicken. Vorsicht: Aus dem Quadranten »herausgescrollte« Punkte gehen verloren! Sie sind dann nur noch durch »UNDO« zu retten, wenn zwischendurch keine andere Operation durchgeführt wurde. **Die Schalter XM, YM, BL und HL**

Mit diesen Schaltern wird die Größe des orangenen Rechtecks sowie die Position von Helpline und Baseline eingestellt. Mit **XM** ändert man die Höhe der Matrix; mit **YM** ihre Breite. **BL** und **HL** verschieben die Position der Base- beziehungsweise Helpline. Um die angezeigten Zahlenwerte zu ändern, klickt man mit der Maus auf den gewünschten Pfeil. Die Pfeilspitze zeigt dabei in die

Richtung, in die sich die jeweilige Linie oder Rechteckbegrenzung verschiebt. Sollen die Zahlenwerte mit der Hand eingegeben werden, so klickt man auf die zu ändernde Zahl und gibt den neuen Wert mit der Tastatur ein.

Unterhalb der Zeichenmatrix befinden sich weitere Befehle des Programms, die wie gewohnt mit der Maus angeklickt werden. Sie beziehen sich nur auf das orange umrandete Rechteck.

MIRR_H

Der Buchstabe wird an der Helpline gespiegelt.

MIRR_B

Der Buchstabe wird an der Baseline gespiegelt.

Rotieren, spiegeln, vergrößern

ROTATE

Der ganze Buchstabe wird um 90 Grad nach links verdreht. Dazu werden innerhalb der Gelb umrandeten Matrix die Zeilen und Spalten vertauscht.

INVERT

Der Buchstabe wird invertiert.

EXPAND_X und EXPAND_Y

Der Buchstabe wird in X- beziehungsweise Y-Richtung vergrößert.

LOAD

Lädt einen Zeichensatz von der Diskette. Beim Anklicken von LOAD erscheint der LOAD-Requester (Bild 3), in dem der gewünschte Zeichensatz zum Laden ausgewählt wird. Das Programm lädt nun automatisch das Verzeichnis »DF0:fonts/« und zeigt es auf dem Bildschirm an. Um einen Zeichensatz von einem anderen Diskettenlaufwerk zu laden, klicken Sie einfach auf »DF1:«, »DF2:« oder »DF3:«. Dann wird das »Fonts«-Verzeichnis des gewählten Laufwerks angezeigt. Mit Hilfe

der beiden Pfeile wird ein Verzeichnis hinauf- und heruntergescrollt, falls es nicht vollständig in das Ausgabefenster paßt. Wollen Sie aus einem Unterverzeichnis in das übergeordnete (also eine Ebene höher liegendes) Verzeichnis springen, klicken Sie auf »PARENT«. Um beliebige Verzeichnis- und Device-Namen eingeben zu können, klickt man das Feld rechts neben »Drawer« mit der Maus an und gibt den gewünschten Namen per Tastatur ein.

Wie Sie sicher wissen, befinden sich im Verzeichnis »fonts« für jeden Zeichensatz ein Unterverzeichnis (Beispiel: »ruby«) und eine Steuerdatei (Beispiel: »ruby.font«). Der Fontdesigner stellt Unterverzeichnisse in Gelb und Dateien in Schwarz dar. Wichtig für das Laden von Zeichensätzen sind nur die Unterverzeichnisse! Wollen Sie also zum Beispiel von Ihrer Workbench-Diskette den Zeichensatz »ruby« laden, so klicken Sie auf das Unterverzeichnis »ruby«. Sie sehen dann in schwarzer Farbe die von diesem Zeichensatz vorhandenen Font-Höhen. Nach einem Klick auf die gewünschte Höhe und auf »OK« wird der ausgewählte Zeichensatz geladen. Eines sollten Sie allerdings beachten: Zeichensätze mit einer Höhe von mehr als 50

kann der Fontdesigner nicht bearbeiten.

Beschleunigtes Directory

Da der Amiga beim Laden von Verzeichnissen bekanntermaßen nicht gerade schnell ist, läßt sich mit einem Klick auf »FAST« ein Fast-Directory im momentan ausgewählten Verzeichnis erzeugen. Besonders interessant ist das natürlich für die »fonts«-Verzeichnisse Ihrer Disketten. Beim Speichern eines Zeichensatzes wird das Fast-Directory im Verzeichnis »fonts« daher auch automatisch auf den neuesten Stand gebracht. Verändern Sie andere Verzeichnisse, so ist das Fast-Directory mit einem weiteren Klick auf »FAST« zu aktualisieren.

Wenn Sie Ihre Zeichensätze nicht nach der auf dem Amiga üblichen Methode (also im Unterverzeichnis »fonts«) abspeichern, beachten Sie bitte die Hinweise in der Tabelle 3.

Sie müssen nicht immer den gesamten Zeichensatz laden. Geben Sie bei »FROM ASCII« den Bereichsbeginn und bei »TO ASCII« das Bereichsende an. Die Eingaben dabei beziehen sich auf die Nummer des ASCII-Zeichens (65 entspricht beispielsweise dem großen »A«, 98 dem kleinen »a«).

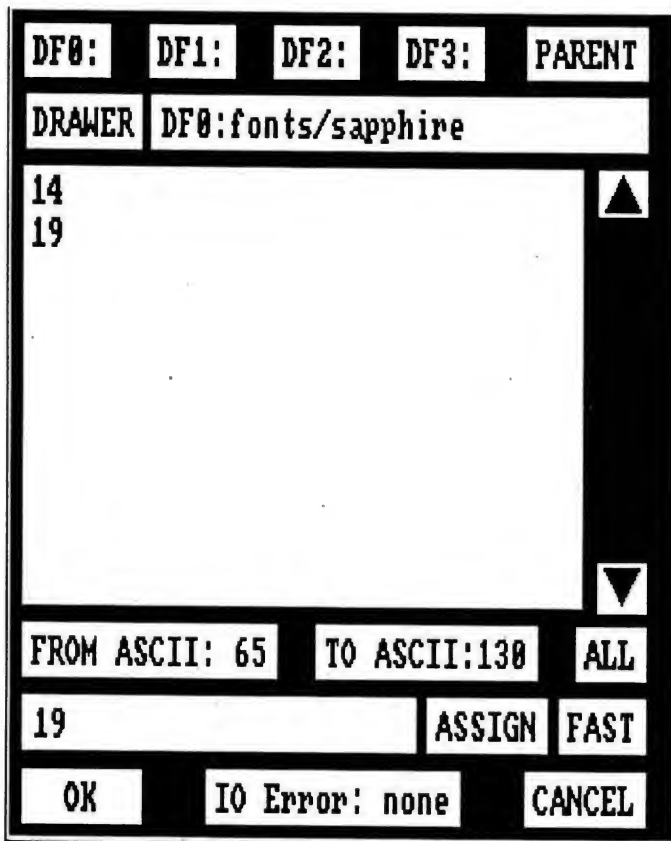


Bild 3. Der LOAD-Requester des Programms

Streng genommen läßt sich auch ein einzelnes Zeichen aus einem Zeichensatz laden. Der restliche Zeichensatz im Speicher wird dabei nicht gelöscht. Wählen Sie ALL, wird der gesamte Zeichensatz geladen.

Um den angewählten Zeichensatz zu laden, klickt man schließlich »OK« an; »CANCEL« dient zum Abbruch der Funktion.

SAVE

Zeichensatz abspeichern. Wie schon bei LOAD öffnet sich ein weiteres Fenster, in dem verschiedene Eingaben erfolgen können:

1. Der vollständige Font-Name, also der Pfadname, der zu diesem Font führt.

2. »HEIGHT« ist die Font-Höhe. Es wird, falls Sie die Höhe nicht ändern, automatisch der Wert für YM (Y-Matrix) benutzt.

3. »LO CHAR« und »HIGH CHAR«: Nicht jeder Zeichensatz muß alle ASCII-Codes ausfüllen. In diesen beiden Feldern steht der erste »LO« beziehungsweise der letzte »HIGH«-ASCII-Code, der zu diesem Font gehört. Verändern Sie diese Werte — falls nötig — durch einfaches Anklicken mit der Maus und anschließender Eingabe des Wertes über die Tastatur.

Proportional oder nicht?

4. »PROPORTIONAL« und »FIXED WIDTH«: Bei einem Proportionalzeichensatz kann jeder Buchstabe eine individuelle Breite besitzen (sie wird vom Programm beim Speichern automatisch berechnet); »Fixed-Width«-Zeichensätze haben dagegen bei allen Zeichen die Breite, die bei »XM« eingestellt wurde. Wählen Sie also mit der Maus, wie Ihr Zeichensatz abgespeichert werden soll.

5. »OK«: Zeichensatz speichern. Da der Speichervorgang vor allem bei größeren Zeichensätzen einige Minuten dauern kann, werden während des Speicherns in der linken oberen Ecke des Bildschirms einige Zwischenmeldungen angezeigt:

»Writing xxxx.font«: Die »font«-Datei für den jeweiligen Zeichensatz wird erzeugt.

»Examining Font«: Interne Berechnungen werden durchgeführt.

»Writing char data«: Der eigentliche Zeichensatz wird geschrieben.

»Creating new Fast-Dir«: Das

Fast-Directory wird (falls eines vorhanden war) aktualisiert.

Falls Sie die Meldung »Error! Font not saved!« bekommen sollten, haben Sie wahrscheinlich versucht, einen Zeichensatz in ein nicht existierendes Verzeichnis zu speichern.

6. »GET NAME«: Name wie unter LOAD beschrieben auswählen.

7. »CANCEL«: Zeichensatz nicht speichern.

Mehr Komfort mit UNDO

UNDO

Dieser Befehl macht die letzte Operation rückgängig.

MOVE

Verschiebung von Zeichen innerhalb eines Zeichensatzes. Sie müssen bei MOVE drei Zahlenwerte in das geöffnete Fenster eingeben:

1. Bei »FROM ASCII«: den Anfang des Bereichs, der verschoben werden soll.

2. Bei »UNTIL ASCII«: das Ende dieses Bereichs.

3. Bei »TO ASCII«: den Anfang des Bereichs, an den verschoben werden soll.

Beispiel: Sie möchten die Zeichen mit den ASCII-Codes 65 bis 90 nach 128 verschieben. Dann ist FROM ASCII=65; UNTIL ASCII=90; TO ASCII=128.

PROGRAMM

Dies ist eine der interessantesten Funktionen von »Font-Designer«! Möchten Sie bei einem Zeichensatz mehrere ASCII-Codes auf die gleiche Art und Weise verändern, können Sie das getrost dem Amiga überlassen. Geben Sie dazu in den entsprechenden Feldern den ersten (»START«) und letzten ASCII-Code (»END«) sowie die Schrittweite (»STEP«) an. Nun werden die einzelnen Funktionen ausgewählt, die der Amiga mit dem gewählten Zeichenbereich durchführen soll.

Die Aufschrift der Schalter entspricht in etwa der Aufschrift im Hauptdisplay (Bild 4). »Q1« bis »Q4« bezeichnen die Quadrantenauswahl für das Scrollen. Dabei steht »Q1« für oben links, »Q2« für oben rechts, »Q3« für unten links und »Q4« für unten rechts. Das »Programm« wird einfach mit Hilfe von Mausclicks auf die gewünschten Felder eingegeben. Sollten Sie sich »vertippt« haben, so läßt sich das eingegebene Programm durch einen Klick auf »PZ« löschen. Schließlich wird der Vorgang mit »GO« gestartet — mit die-

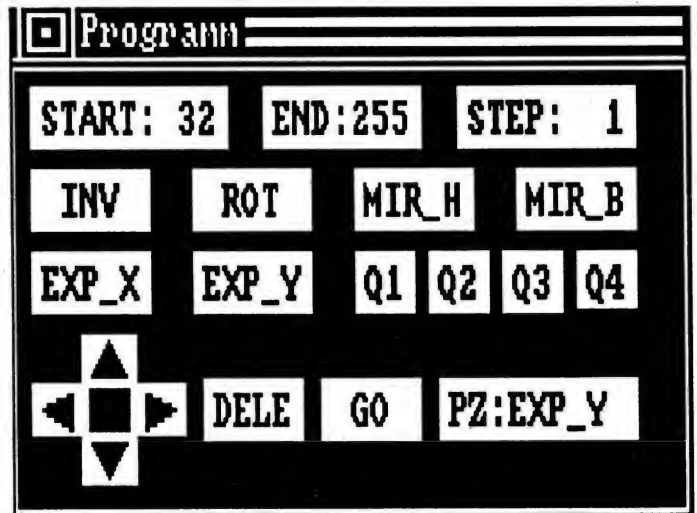


Bild 4. Durch Auswahl im »PROGRAMM«-Fenster wird vieles automatisch erledigt.

sem Schalter kann das Programm auch wieder abgebrochen werden. Rechts unten befindet sich der PZ (Programmzähler), in dem der zuletzt eingegebene Programmschritt angezeigt wird.

Die Programm-Automatik

Ein Beispiel: Sie wollen sämtliche kleinen Buchstaben ihres Zeichensatzes nach links drehen. Dann geben Sie nach dem Klick auf »PROGRAMM« bei »START« 97 und bei »END« 123 ein. Nun klicken Sie einmal auf »ROT«. Daraufhin erscheint bei »PZ« der gerade eingegebene Programmschritt »ROT«. Nun starten Sie das Ganze mit »GO«. Rechts unten neben dem Programmfenster wird der ASCII-Code des gerade

bearbeiteten Zeichens angezeigt.

Verlassen Sie dieses Fenster durch Betätigen des Schließ-Gadgets oben links.

SHOW

Zeigen des gesamten Zeichensatzes. Von jedem Zeichen wird der durch das orange Rechteck umgrenzte Bereich in Originalgröße angezeigt. Sollten alle Zeichen nicht auf einmal auf den Bildschirm passen, wird mit der rechten Maustaste weitergeschaltet. Die linke Maustaste beendet die Ausgabe.

Um sich mit den vielen Fähigkeiten des Fontdesigners vertraut zu machen, brauchen Sie einfach ein bißchen Übung. Ein Vorschlag: Versuchen Sie doch einmal, Ihre Lieblingszeichensätze mit deutschen Sonderzeichen zu versehen.

(Richard Aretz/
Andreas Lietz/rs)

- Zeichensätze sind bis zu einer Größe von 64 x 50 Punkten zu bearbeiten.
- Das angewählte Zeichen wird in einer Matrix vergrößert und in Originalgröße dargestellt. Um es zu verändern, »malt« man mit der Maus in die Matrix.
- Ein Zeichen ist in einen Zwischenspeicher zu kopieren, um es an anderer Stelle zu verwenden.
- Zeichen invertieren, spiegeln, drehen, vergrößern, scrollen.
- Mit UNDO Befehle rückgängig machen.
- Mit Hilfe eines Programm-Modus diese Veränderungen bei vielen Zeichen automatisch durchführen.
- Alle Zeichen auf einmal in Originalgröße anzeigen.
- Zeichen innerhalb eines Zeichensatzes bereichsweise verschieben.
- Standard-Amiga-Zeichensätze laden und speichern.
- Zeichensätze mit Proportionalchrift oder gleichbleibender Breite speichern.

Tabelle 1. Was kann der Fontdesigner?
Die Kurzübersicht zeigt alle Vorzüge des Programms.

COMPILIEREN:

```
cc h0.c +l -s
cc h1.c +l -s
cc h2.c +l -s
cc h3.c +l -s
cc fontdesigner +l -s
```

LINKEN:

```
ln fontdesigner.o h0.o h1.o h2.o h3.o -lm32 -lc32
```

Tabelle 2a. Die Compiler- und Linker-Anweisungen für den Aztek-C-Compiler, Version 3.4a

COMPILIEREN:

```
lc h0
lc h1
lc h2
lc h3
lc fontdesigner
```

LINKEN:

```
blink LIB:c.o + fontdesigner.o + h0.o + h1.o + h2.o +
h3.o LIB LIB:amiga.lib + LIB: m.lib + LIB: c.lib
```

Tabelle 2b. Die Anweisungen für den Lattice-C-Compiler

Unter normalen Umständen ist es das beste, wenn Sie Ihre Zeichensätze immer in einem Verzeichnis mit dem Namen »fonts« verwalten. Sollten Sie sie aber in ein anderes Verzeichnis speichern wollen (z.B. bei Verwendung einer Festplatte), dann beachten Sie bitte, daß der Amiga beim Laden von Diskettenfonts den Befehl »Assign fonts: newdir:« benötigt, wobei »newdir« für den Namen Ihres Fontverzeichnis steht. Der »Assign«-Befehl wird normalerweise von Fontdesigner automatisch bei der Auswahl eines anderen Diskettenlaufwerks für das Verzeichnis »fonts« dieser Diskette ausgeführt.

Um den Befehl auf Ihr Zeichensatzverzeichnis manuell anzuwenden, klicken Sie im LOAD-Requester einfach auf »ASSIGN«, sobald Sie Ihr Zeichensatzverzeichnis ausgewählt haben. Dann können Sie den gewünschten Font-Namen anklicken, sich in dem nun erscheinenden neuen Verzeichnis eine Font-Höhe aussuchen und den Zeichensatz laden.

Falls Sie den Klick auf »ASSIGN« vergessen sollten, erscheint die Meldung »I can't find this font«, und der Zeichensatz »topaz 8« wird automatisch geladen. Sollte Ihnen das passieren, klicken Sie im LOAD-Requester auf »PARENT«, anschließend auf »ASSIGN« und dann auf den Namen Ihres Zeichensatzes. Nach dem Klick auf »OK« sollten nun keine Probleme mehr auftreten.

Tabelle 3. Wichtige Hilfestellung bei Problemen mit dem Fontdesigner

Programmname: »Fontdesigner«

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Compiler: Aztec-C V3.4a / Lattice-C

Aufrufe: siehe Tabelle 2a bzw. 2b

Bemerkung: siehe Tabelle 3 sowie Text

Programm : FontDesigner

```
1 Hy0 /*---(FontDesigner)----- File test.c (Hauptprogramm) -----
2 H6 #include "header.h"
3 82 /*----- Globale Variablen -----
4 X0 struct IntuitionBase *IntuitionBase;
5 rU struct GfxBase *GfxBase;
6 c1 ULONG DiskFontBase,DosBase;
7 K9 long XMatrix=8,YMatrix=8,BaseLine=7,HelpLine=4,DMode=1,Ascii
8 Kd long Quadrant=1,undoascii=65;
9 Az struct TextFont *tftopaz,*tfuser;
10 I4 struct Window *wind;
11 ht struct RastPort *rast;
12 xK ULONG *omatrix;
13 wt BYTE *drawmem;
14 h6 struct Image *oima,*drima,*mima;
15 Pl struct FileInfoBlock *f_info;
16 qh ULONG maske[]=
17 Dg {
18 mC3 0x80000000,1073741824,536870912,268435456,134217728,67108
19 Gr 33554432,16777216,8388608,4194304,2097152,1048576,524288,
20 I7 262144,131072,65536,32768,16384,8192,4096,2048,1024,512,2
21 r2 64,32,16,8,4,2,1,
22 Fb0 };
23 Gr ULONG cmemo[100],scmemo[100],undomem[100];
24 ev extern short readat[][4],Fascii,Tascii;
```

```
25 jV /*----- HAUPTPROGRAMM -----
26 7T main()
27 Nq {
28 DX ULONG class;
29 LY USHORT code;
30 jB short mox,moy,i,mode,j;
31 uK long *adr,*point;
32 CT BYTE flag=0;
33 vA3 Oplibs();
34 nB if(GetMemory(1)<0)
35 Vy {
36 Qf6 Shure(3);
37 PO Cllibs();
38 jR exit(0);
39 d83 }
40 q1 if(Display()<0)
41 b4 {
42 W16 Shure(3);
43 VU Cllibs();
44 HF GetMemory(0);
45 AY exit(0);
46 kF3 }
47 Of tfuser=tftopaz=rast->Font;
48 K1 FOREVER
49 jC {
50 Hn6 Wait(1<<wind->UserPort->mp_SigBit);
51 SX while(ExaIntui(wind,&class,&code))
52 mF {
53 E89 mox=wind->MouseX;moy=wind->MouseY;
54 Wo switch (class)
55 pI {
56 KWC case CLOSEWINDOW:
57 LmF if(Shure(0)==1)break;
58 bS if(flag==1){ RemFont(tfuser);CloseFont(tfuser
59 JI ); }
60 XV CloseWindow(wind);
61 nm GetMemory(0);
62 Rp Cllibs();
63 F4C exit(0);
64 IkF case VANILLAKEY:
65 AJ adr=cmatrix+100*Ascii;point=undomem;
66 ZH undoascii=Ascii;
67 k4 for(j=0;j<=99;j++,point++,adr++)*point=*adr;
68 k4 Ascii=code;
```



```

68 tm      AsciiOut();
69 CQ      Mem_to_Matrix();
70 3C      break;
71 bzC     case MOUSEBUTTONS:
72 wu      if(code!=SELECTDOWN && code!=MENUDOWN)break;
73 6f      mode=1;
74 d9      if(code==MENUDOWN)mode=0;
75 X5      i=Prurec(mox,moy,recdat,0,40);
76 3t      if(i<0)break;
77 kI      if(i>=4 && i<=15){ ChangeMatrix(i);break; }
78 Pn      if(i != 32)
79 Dg      {
80 YOF      adr=cmatrix+100*Ascii;point=undomem;
81 Qz      undoascii=Ascii;
82 pX      for(j=0;j<=99;j++,point++,adr++)*point=*adr;

83 LqC     }
84 mh      if(i>=29)Invert(rast,recdat[i]);
85 pf      switch (i)
86 Kn      {
87 WpF      case 0: DrawInto(mode);break;
88 XV      case 1: case 2: case 3:
89 nmI      ChangeAscii(i-1);
90 NW      break;
91 zoF      case 16:
92 RgI      if(DMode==1)break;
93 bO      DMode=1;
94 79      Invert(rast,recdat[16]);Invert(rast,recdat
[17]);
95 Sb      break;
96 7lF      case 17:
97 UiI      if(DMode==0)break;
98 dI      DMode=0;
99 CE      Invert(rast,recdat[16]);Invert(rast,recdat
[17]);
break;
case 18: case 19: case 20:
Invert(rast,recdat[i]);
Delete(i-18);
Invert (rast,recdat[i]);
break;
case 21: case 22: case 23: case 24:
Invert(rast,recdat[i]);
Scroll(i-21);
Invert(rast,recdat[i]);
break;
case 25: case 26: case 27: case 28:
if((i-25)==Quadrant)break;
Invert(rast,recdat[Quadrant+25]);
Quadrant=i-25;
Invert(rast,recdat[i]);
break;
case 29: case 30: Mirror(i-29);break;
case 31: Rotate();break;
case 32: Undo();break;
case 33: Revers();break;
case 34:
if(flag==1){ RemFont(tfuser);CloseFont(tfu
ser); }
flag=1;
j=LoadDiskFont();
if(j==3){ flag=0;Shure(3);break; }
if(j==2){ flag=0;break; }
if(j==1)
{
tfuser=tfstopaz;
Shure(1);
flag=0;
}
if(flag==1)Shure(6);
else Shure(7);
for(j=FAscii;j<=TAscii;j++)
Font_to_Mem(j);
Shure(6);
Ascii=65;
XMatrix=tfuser->tf_XSize;
if(XMatrix > 64)XMatrix=64;
YMatrix=tfuser->tf_YSize;
if(YMatrix > 50)YMatrix=50;
BaseLine=tfuser->tf_Baseline;
if(BaseLine > 50)BaseLine=49;
HelpLine=XMatrix/2;
Mem_to_Matrix();

```

```

147 cn      AsciiOut();Matout();
148 JS      break;
149 nAF      case 35: case 36:
150 IrI      if(i==35) Expand_x();
151 2g      else Expand_y();
152 NW      break;
153 oYF      case 37: SaveFont();break;
154 kN      case 38: Programm();break;
155 NZ      case 39: Show();break;
156 uu      case 40: MoveAscii();break;
157 X2C      }
158 yt      if(i>=29)Invert(rast,recdat[i]);
159 Z49      }
160 a56      }
161 b63      }
162 c70      }
(C) 1987 M&T

```

Listing 1. Das Programm »fontdesigner« ist in der Sprache C geschrieben. Geben Sie das Listing bitte mit dem Checksummer (Seite 159) ein. Beachten Sie bitte die Compiler- und Linker-Anweisungen in Tabelle 2a beziehungsweise 2b.

Programmname: h0.c

Computer: s. Listing 1

Programmname: h0.c

```

1 LJ0 /* --(FontDesigner)----- File h0.c ----- */
2 9E #include <exec/memory.h>
3 Rg #include <intuition/intuition.h>
4 L9 #include <intuition/intuitionbase.h>
5 3q #include <libraries/dos.h>
6 15 #include <graphics/gfxmacros.h>
7 MB #include "header.h"
8 aN /*----- Externe Variablen aus main -----
-----*/
9 TS extern struct TextFont *tfuser,*tftopaz;
10 48 extern short *drawmem;
11 18 extern long XMatrix,YMatrix,BaseLine,Ascii;
12 Az extern ULONG *cmatrix;
13 nb extern struct FileInfoBlock *f_info;
14 fI extern struct RastPort *rast;
15 JL extern struct Image *oima,*mima;
16 GY /*----- Daten für GetFilename() -----
-----*/
17 Xo short fdat[][4]=
18 IB3 { 10,16,48,28 },
19 9b { 69,16,107,28 },
20 yI { 128,16,166,28 },
21 I3 { 187,16,225,28 },
22 2X { 248,16,302,28 },
23 Mb { 70,33,302,45 },
24 kC { 10,50,272,152 },
25 Bg { 282,50,302,60 },
26 jL { 282,142,302,153 },
27 PO { 10,174,194,186 },
28 gA { 10,191,64,203 },
29 DI { 248,191,302,203 },
30 Q3 { 264,174,302,186 },
31 E1 { 202,174,256,186 },
32 Sa { 10,157,128,169 },
33 4P { 149,157,251,169 },
34 ZN { 272,157,302,169 },
35 c8 { 97,191,215,203 },
36 tpO };
37 8n char *ftxt[]=
38 cc3 "DF0:", "DF1:", "DF2:", "DF3:", "PARENT", "DRAWER", "", "", "",
",", " OK",
39 NH "CANCEL", "FAST", "ASSIGN", "FROM ASCII:", "TO ASCII:", "ALL",
",
40 dp "IO Error: none",
41 yuO };
42 aI short fpfeil[][7]=
43 uI3 { 292,51,284,59,300,59,-2 },

```

Listing 2. »h0.c« bitte mit dem Checksummer eingeben


```

44 w9      [ 292,151,284,143,300,143,-2 ],
45 2y0    ];
46 w1      struct StringInfo gf_info1,gf_info2;
47 nF      struct Gadget gf_gad2=
48 D83     NULL,14,177,178,9,GADGHCOMP,NULL,STRGADGET,
49 iF      NULL,NULL,NULL,NULL,&gf_info2,1,NULL,
50 730    ];
51 pG      struct Gadget gf_gad1=
52 NR3     &gf_gad2,74,36,228,9,GADGHCOMP,RELVERIFY,STRGADGET,
53 hD      NULL,NULL,NULL,NULL,&gf_info1,1,NULL,
54 B70    ];
55 0a      char edrawer[100]="DF0:fonts";
56 zA      char efname[100];
57 tT      short Fascii=0,Fascii=255;
58 rI      /*----- Daten für SaveFont() -----
          -----*/
59 YM      unsigned long ublock1[]=
60 Cy3     0x3f3,0,1,0,0,0x35e,0x3e9,0x35e,0x70004e75,0,0,0x0c0000
          00,
61 39      0x001a0f80,0x00010000,0,0,0,0,0,0,0,0,0,0x0c00,0x1a,0,
62 z0      0,0x0062000f,0x00060001,0,0x6e,0x066e0000,0x9f20000,0xbb
          40000
63 KGO    ];
64 fU      unsigned long ublock2[]=
65 3W3     0,0x03ec0000,0x00060000,0x0,0x000e0000,0x00440000,0x005
          c0000,
66 Th      0x00620000,0x00660000,0x006a0000,0,0x03f20000,
67 OK0    ];
68 zT      short sdat[][4]=
69 t43     { 10,16,64,28 },
70 Zs      { 74,16,300,28 },
71 ID      { 10,33,88,45 },
72 VF      { 88,33,190,45 },
73 6M      { 190,33,300,45 },
74 XY      { 10,50,112,62 },
75 pm      { 206,50,300,62 },
76 Wt      { 10,67,64,79 },
77 R9      { 246,67,300,79 },
78 k4      { 120,67,190,79 },
79 aW0    ];
80 SK      char *stxt[]=
81 wK3     "DRAWER","", "HIGHT:", "LOW CHAR:", "HIGH CHAR:", "PROPORTI
          ONAL",
82 5r      "FIXED WIDTH", " OK", "CANCEL", "GET NAME",
83 ea0    ];
84 Vn      struct StringInfo s_info;
85 pE      struct Gadget s_gad=
86 f63     NULL,78,19,218,9,GADGHCOMP,NULL,STRGADGET,
87 UO      NULL,NULL,NULL,NULL,&s_info,1,NULL,
88 jf0    ];
89 te      char fontbreit[256];
90 lh      /*#0.1----- Diskfont laden -----
          -----*/
91 oe      BYTE LoadDiskFont()
92 f1      struct TextAttr txtatr;
93 5K      BYTE erfolg,hmem[100];
94 DJ      long i;
95 x7      double atof();
96 mQ      static BYTE count=0;
97 uJ3     if(count==0)
98 L86     strcpy(efname,"");
99 fH      strcpy(edrawer,"DF0:fonts");
100 8U      count++;
101 ss3     erfolg=GetFilename(edrawer,efname);
102 ry      if(erfolg < 0) return(erfolg);
103 gE      strcpy(hmem,edrawer);
104 OX      strcpy(edrawer+strlen(edrawer),".font");
105 uF      txtatr.ta_Name=edrawer;
106 Gw      txtatr.ta_YSIZE=(UWORD)atof(efname);
107 HT      txtatr.ta_Style=0;
108 tN      txtatr.ta_Flags=0;
109 up      tfuser=OpenDiskFont(&txtatr);
110 k5      i=0;
111 Ww      if(tfuser==0)
112 UP6     tfuser=ttftopaz;
113 Pn      i=-1;
114 s63     edrawer[strlen(edrawer)-5]=0;
115 Oq      return(i);
116 OS0     /*#0.2----- Filename Holen -----
          -----*/
117 Iw      BYTE GetFilename(drawer,filename)
118 G8      char filename[],drawer[];
119 fF      ULONG flags=WINDOWDRAG|WINDOWCLOSE|WINDOWDEPTH|SMART REF

```

```

SH]
120 JFC          ACTIVATE;
121 D10          ULONG iflags=MOUSEBUTTONS] CLOSEWINDOW] GADGETUP] GADGETDOWN]
DISKINSERTED;
122 xd          struct Window *wn;
123 HX          struct RastPort *rp;
124 PV          BYTE i,offs=3,*names,*point,flag,mode=0;
125 KY          short mox,moy;
126 n7          ULONG class;
127 v8          USHORT code;
128 81          short fzah1,start=1,nr,aktdrive=0;
129 EP          short DirScroll();
130 wJ3          wn=Make_Window(20,10,320,209,"GET_FILENAME",flags,iflag
s,0);
131 F1          if(wn==0) return(-3);
132 eg          names=AllocMem(1024,MEMF_CLEAR);
133 pm          if(names==0) return(-3);
134 Bv          rp=wn->RPort;
135 lc          SetAPen(rp,1);RectFill(rp,0,11,320,209);
136 9d          Box(rp,2,12,317,207,0,JAM1);
137 IJ          for(i=0;i<=17;i++)
138 J96          {
139 U1          {
SetAPen(rp,0);
RectFill(rp,fdat[i][0],fdat[i][1],fdat[i][2],fdat[i]
[3]);
Box(rp,fdat[i][0],fdat[i][1],fdat[i][2],fdat[i][3],2
,JAM1);
141 K1          Box(rp,fdat[i][0]-1,fdat[i][1],fdat[i][2]+1,fdat[i][
3],2,JAM1);
142 mT          if(i==5)
143 LD9          {
offs=-57;SetAPen(rp,0);
144 L5          RectFill(rp,10,33,64,45);
145 Z26          else offs=3;
146 Hq          TextOut(rp,fdat[i][0]+offs,fdat[i][1]+3,3,0,JAM2,ft
xt[i],0);
147 qY3          FDrawPolygon(rp,fpfeil[0],3,JAM2);
148 wf          FDrawPolygon(rp,fpfeil[1],3,JAM2);
149 XF          gf_info1.Buffer=drawer;gf_info2.Buffer=filename;
150 a1          gf_info1.UndoBuffer=NULL;gf_info2.UndoBuffer=NULL;
151 RD          gf_info1.MaxChars=100;gf_info2.MaxChars=100;
152 b0          AddGadget(wn,&gf_gad1,-1);AddGadget(wn,&gf_gad2,-1);
153 Pr          RefreshGadgets(&gf_gad1,wn,NULL);
154 Or          fzah1=GetDir(names,drawer,mode);
155 qN          DirOut(rp,names,start,fzah1);
156 41          FOREVER
157 ww6          while (ExaIntui(wn,&class,&code))
158 ZJ9          {
mox=wn->MouseX;moy=wn->MouseY;
159 DV          switch (class)
160 OCC          {
case CLOSEWINDOW:
161 IuF          flag=-2;
162 5zC          end:
163 3vF          FreeMem(names,1024);
164 zx          RemoveGadget(wn,&gf_gad1);RemoveGadget(wn,&
gf_gad2);
CloseWindow(wn);
return(flag);
case DISKINSERTED:
165 FZ          strcpy(drawer,ftxt[aktdrive]);
166 BK          strcpy(&drawer[4],"fonts");
167 o1C          Assign(drawer);
case GADGETUP: case GADGETDOWN:
168 zHF          RefreshGadgets(&gf_gad1,wn,NULL);
169 a1          fzah1=GetDir(names,drawer,mode);
170 uZ          start=1;
171 dAC          DirOut(rp,names,start,fzah1);
172 iAF          break;
case MOUSEBUTTONS:
173 JA          if(code != SELECTDOWN)break;
174 zo          i=Prurec(mox,moy,fdat,0,16);
175 Ah          if(i<0)break;
176 lu          if(i!=6 && i<14)Invert(rp,fdat[i]);
177 JhC          switch(i)
178 bpF          {
case 0: case 1: case 2: case 3: case 4:
case 12:
179 zS          if(i!=4 && i!=12)
180 jZ          aktdrive=i;
181 cq          strcpy(drawer,ftxt[i]);
182 wV          strcpy(&drawer[4],"fonts");
183 AeI          Assign(drawer);
else
184 mpL          if(i==4)
185 n5O          Parent(drawer);
186 GR          if(i==12)mode=1;
187 s3          fzah1=GetDir(names,drawer,mode);
188 Cr          Assign(drawer);
189 sfL          else
190 VBO          if(i==4)
191 LWQ          Parent(drawer);
192 IXL          if(i==12)mode=1;
193 dU          fzah1=GetDir(names,drawer,mode);

```



```

194 lZ      mode=0;
195 K9      start=1;
196 V2      DirOut(rp,names,start,fzahl);
197 cd      Invert(rp,fdat[1]);
198 7C      break;
199 PlI
200 pQL
201 OF      case 6:
202 RC      nr=(moy-52)/10+start;
203 vq      if(nr>=fzahl)break;
204 9U      i=52+(nr-start)*10;
205 mJ      SetDrMd(rp,COMPLEMENT);
206 dY      RectFill(rp,11,i,271,i+7);
207 5K      Delay(5);RectFill(rp,11,i,271,i+7);
208 WtO      point=names;
209 cB      for(i=1;i<=nr-1;i++)
210 gKL      point++;
211 uY      point+=strlen(point)+1;
212 ToO      moy=*point;point++;
213 yKR      if(moy==1)
214 qJO      if(drawer[strlen(drawer)-1] != ':')
215 zq      strcpy(drawer+strlen(drawer),"\\0");
216 fU      strcpy(drawer+strlen(drawer),point);
217 qN      fzahl=GetDir(names,drawer,mode);
218 I2L      start=1;
219 Sb      DirOut(rp,names,start,fzahl);
220 wMI      else strcpy(filename,point);
221 aEL      break;
222 lZ      case 7: case 8:
223 Wf      start=DirScroll(wn,names,start,fzahl,1);
224 LeI      Invert(rp,fdat[1]);
225 aQL      break;
226 AZ      case 10: case 11:
227 ef      if(i==10) flag=0;
228 92      else flag=-2;
229 4eI      Delay(5);Invert(rp,fdat[i]);
230 FML      goto ende;
231 tY      case 13:
232 HN      ModifyIDCMP(wn,wn->IDCMPFlags&(&BDISK
233 CD      INSERTED));
234 hq      Assign(drawer);
235 DoI      ModifyIDCMP(wn,wn->IDCMPFlags|DISKIN
236 VIL      SERTED);
237 Z8      Invert(rp,fdat[1]);
238 Plk      break;
239 awL      case 14:
240 nw      SetAPen(rp,3);SetDrMd(rp,JAM2);
241 MyI      Fascii=(short)input_zahl(wn,fdat[14][
242 bOL      0]+91,
243 L6      fdat[14][1]+
244 Yvn      3,3,"");
245 vML      if(Fascii<0 || Fascii>Tascii)Fascii
246 t2      =0;
247 V8I      break;
248 RSL      case 15:
249 UR      SetAPen(rp,3);SetDrMd(rp,JAM2);
250 Kb      Tascii=(short)input_zahl(wn,fdat[15][
251 Jb      0]+75,
252 z8      fdat[15][
253 lTF      1]+3,3,"");
254 9c9      if(Tascii>255 || Tascii<Fascii)Tasc
255 qbO      ii=255;
256 45      break;
257 8o      case 16:
258 mQ      Invert(rp,fdat[1]);
259 3U      Fascii=0;Tascii=255;
260 xH      Delay(5);
261 5I      Invert(rp,fdat[16]);
262 wE3      break;
263 nU      RefreshGadgets(&gf_gad1,wn,NULL);
264      ErrOut(rp);
265      /*--- Unterfunktion zu h0.2 ---*/
266      short DirScroll(wn,names,start,fzahl,mode)
267      struct Window *wn;
268      char *names;
269      short start,fzahl,mode;
270      ULONG class;
271      USHORT code;
272      Delay(10);
273      FOREVER

```

```

264 vf6      switch (mode)
265 Vs9      case 7:
266 DpC      if(start>1)
267 mHF      start--;
268 lK      DirOut(wn->RPort,names,start,fzahl);
269 GPC      break;
270 cO9      case 8:
271 oIC      if(start<(fzahl-10))
272 h8F      start++;
273 6P      DirOut(wn->RPort,names,start,fzahl);
274 LUC      break;
275 Um6      if(ExaIntui(wn,&class,&code))break;
276 RS3      return(start);
277 VZO      /*h0.3----- Directory laden -----*/
278 LR      -----*/
279 ZS      short GetDir(namen,dirname,mode)
280 oX      BYTE mode;
281 hc      char *namen,*dirname;
282 Tc      struct FileLock *lockp;
283 qm      register short anz;
284 Ln      short flag;
285 p93      char *point=namen;
286 fG6      if((lockp=Lock(dirname,ACCESS_READ))==0)
287 he3      return(-1);
288 g4      if(Examine(lockp,f_info)==0)return(-1);
289 Kd      if(f_info->fib_DirEntryType<0) return(-1);
290 co      flag=FastDir(0,dirname,point,anz);
291 S16      if(flag>0 && mode==0)
292 DM      SortDir(point,flag);
293 y23      return(flag);
294 8o6      while(ExNext(lockp,f_info)==-1)
295 4k9      if(f_info->fib_DirEntryType<0)
296 bO6      *namen=0;
297 Ar9      else
298 Bb6      *namen=1;
299 Ym      namen++;
300 5v      strcpy(namen,f_info->fib_FileName);
301 M1      namen+=(strlen(f_info->fib_FileName)+1);
302 f23      anz++;
303 j2      UnLock(lockp);
304 lU6      if(IoErr()==ERROR_NO_MORE_ENTRIES)
305 Jf      if(mode==1) FastDir(1,dirname,point,anz);
306 pA      SortDir(point,anz);
307 pD3      return(anz);
308 FsO      else return(-1);
309 JB      /*h0.4----- Directory ausgeben -----*/
310 cG      -----*/
311 JZ      void DirOut(rp,names,start,max)
312 XG      char *names;
313 zr      struct RastPort *rp;
314 5F      short start,max;
315 YH3      short xa=13,ya=52,i,anz=max-start;
316 F3      char farbe;
317 zf      SetAPen(rp,0);SetDrMd(rp,JAM1);
318 z56      RectFill(rp,11,51,271,151);
319 y4      for(i=1;i<=start-1;i++)
320 JT3      names++;
321 ET      names+=strlen(names)+1;
322 Rk6      if(anz>10)anz=10;
323 wC      for(i=1;i<=anz;i++,ya+=10)
324 Wq9      farbe=2;
325 6C6      if(*names==1)
326 3S      farbe=3;
327 6C      names++;
328 eqO      Text_out(rp,xa,ya,farbe,0,JAM2,names,0);
329 HF      names+=strlen(names)+1;
330 cs      /*h0.5----- Io Fehler ausgeben -----*/
331 28      -----*/
332 2A      void ErrOut(rp)
333 mw3      struct RastPort *rp;
334 GJ      long err;
335 tB6      char htxt[15];
336 F23      err=IoErr();
337 oY6      if(err<103 || err>231)
338 a13      strcpy(htxt,ftxt[17]);
339 6G      else
340      sprintf(htxt,"IO Error: %3.0f", (float)err);
341      Text_out(rp,fdat[17][0]+3,fdat[17][1]+3,3,0,JAM2,htxt,14);
342      sprintf(htxt,"%3.0f", (float)Fascii);

```

Listing 2. (Fortsetzung)


```

340 wI      Text_out(rp,fdat[14][0]+91,fdat[14][1]+3,3,0,JAM2,htxt,
341 Ge      sprintf(htxt,"%3.0f",(float)TAscii);
342 4e      Text_out(rp,fdat[15][0]+75,fdat[15][1]+3,3,0,JAM2,htxt,
343 q50     /*h0.6----- Fast Directory lesen oder erzeugen --
-----*/
344 Mc      short FastDir(mode,drawer,nam,anz)
345 tx      char mode,*drawer,*nam;
346 dK      short anz;
347 YI      ULONG datei;
348 Xf      char dateiname[80],text[10],*point=nam;
349 26      short bytes=0,i;
350 Cp3      strcpy(dateiname,drawer);
351 LJ      if(drawer[strlen(drawer)-1]!=':')
352 Y76      strcpy(text,"fd.fdir");
353 WJ3      else
354 Ur6      strcpy(text,"/fd.fdir");
355 J93      strcpy(dateiname+strlen(dateiname),text);
356 tW      switch(mode)
357 l16      case 0:
358 2q9      datei=Open(dateiname,MODE_OLDFILE);
359 28      if(datei==0)return(-1);
360 G6      if(Read(datei,&anz,2)!=2) goto abbruch;
361 Bb      Read(datei,nam,1024);
362 lu      break;
363 tA6      case 1:
364 U49      for(i=1;i<=anz-1;i++)
365 3QC      point++;
366 7j      bytes+=strlen(point)+2;
367 AJ      point+=strlen(point)+1;
368 ED9      datei=Open(dateiname,MODE_NEWFILE);
369 CI      if(datei==0)return(-1);
370 dH      if(Write(datei,&anz,2)!=2) goto abbruch;
371 aS      if(Write(datei,nam,bytes)!=bytes) goto abbruch;
372 v4      break;
373 lu3      Close(datei);
374 vG      return(anz);
375 lR0      abbruch:
376 4x3      Close(datei);
377 8j      return(-1);
378 qp0     /*h0.7----- Textfont auf Diskette schreiben ---
-----*/
379 k2      void SaveFont()
380 ec      ULONG flags=ACTIVATE|WINDOWCLOSE|WINDOWDRAG;
381 TA      ULONG iflags=MOUSEBUTTONS|CLOSEWINDOW;
382 vF      ULONG class;
383 3G      USHORT code;
384 eO      long i,help=Ascii;
385 Cs      struct Window *wn;
386 Wm      struct RastPort *rp;
387 9z      short mox,moy,lochar=32,hichar=255;
388 Uh      BYTE proflag=1;
389 SC      void zahl_out();
390 sG3      wn=Make_Window(50,20,318,90,"SAVE FONT",flags,iflags,0)
391 XE      ;
392 L5      if(wn==0)return();
393 XO      rp=wn->RPort;
394 Ih      SetAPen(rp,1);RectFill(rp,0,11,318,90);
395 8u      Box(rp,2,12,315,88,0,JAM1);
396 tJ6      for(i=0;i<=9;i++)
397 DH      SetAPen(rp,0);
398 DD      RectFill(rp,sdat[i][0],sdat[i][1],sdat[i][2],sdat[i]
399 BV      [3]);
400 uq3      Box(rp,sdat[i][0],sdat[i][1],sdat[i][2],sdat[i][3],2
401 mA      ,JAM1);
402 19      Text_out(rp,sdat[i][0]+3,sdat[i][1]+3,3,0,JAM2,stxt[
403 U2      i],0);
404 bC      s_info.Buffer=edrawer;
405 ku      s_info.UndoBuffer=NULL;
406 J7      s_info.MaxChars=100;
407 7o      AddGadget(wn,&s_gad,-1);
408 zz6      RefreshGadgets(&s_gad,wn,NULL);
409 cM9      zahl_out(rp,lochar,hichar);
410 CN      Invert(rp,sdat[5]);
411 3FC      FOREVER
412 71      while (ExaIntui(wn,&class,&code))
413 ROF      mox=wn->MouseX;moy=wn->MouseY;
414 Ga      switch(class)
415 YO      case CLOSEWINDOW:
416 AYC      ende:
417 SgF      RemoveGadget(wn,&s_gad);
418 I8      CloseWindow(wn);
419 aQ      return();
420 K1      case MOUSEBUTTONS:
421 nM      if(code != SELECTDOWN)break;
422 sAI      i=Prurec(mox,moy,sdat,2,9);
423 QpL      if(i<0)break;
424 q2      SetAPen(rp,1);
425 mv      switch(1)
426 yHI      case 2:
427 JxL      YMatrix=(long)input_zahl(wn,61,36,2,"
428 3t      ");
429 qz      if(YMatrix<0 || YMatrix>50) YMatrix
430 4OI      =8;
431 8ZL      break;
432 nu      case 3:
433 u3      lochar=(short)input_zahl(wn,163,36,3,
434 AVI      "");
435 OaL      if(lochar>hichar || lochar<0) locha
436 4N      r=32;
437 Fp      break;
438 z8      case 4:
439 HdI      hichar=(short)input_zahl(wn,273,36,3,
440 1aL      "");
441 4M      if(hichar<lochar || hichar>255) hic
442 Ku      har=255;
443 4D      break;
444 YyH      case 5:
445 2GL      if(proflag==1)break;
446 mV      proflag=1;
447 8PO      Invert(rp,sdat[5]);Invert(rp,sdat[6])
448 1FR      ;
449 tWL      break;
450 Yp      case 6:
451 kd      if(proflag==0)break;
452 DM      proflag=0;
453 bOH      Invert(rp,sdat[5]);Invert(rp,sdat[6])
454 BPI      ;
455 lz      break;
456 xv      case 7: case 8:
457 ES      Invert(rp,sdat[1]);
458 JS      if(i==7)
459 U5C      if(MakeFont(proflag,lochar,hichar)
460 dn      <0)
461 100      Shure(2);
462 9o      Ascii=help;AsciiOut();
463 11      Delay(5);
464 Pu      goto ende;
465 JH      break;
466 dq3      case 9:
467 eE      Invert(rp,sdat[i]);
468 3A      GetFilename(edrawer,efname);
469 Zg      YMatrix=(long)atoi(efname);
470 Vm      Invert(rp,sdat[i]);
471 kt      break;
472 rLO      RefreshGadgets(&s_gad,wn,NULL);
473 40      zahl_out(rp,lochar,hichar);
474 h9      /*---- Unterfunktion zu h0.7 -----*/
475 a5      void zahl_out(rp,lochar,hichar)
476 dq      struct RastPort *rp;
477 H3      short lochar,hichar;
478 OF      char htxt[10];
479 D1      sprintf(htxt,"%2.0f",(float)YMatrix);
480 yJ      Text_out(rp,61,36,1,0,JAM2,htxt,0);
481 Nm      Text_out(rp,163,36,1,0,JAM2,htxt,0);
482 JM      Text_out(rp,273,36,1,0,JAM2,htxt,0);
483 Ip      Text_out(rp,61,36,1,0,JAM2,htxt,0);
484 tD      Text_out(rp,163,36,1,0,JAM2,htxt,0);
485 RO      Text_out(rp,273,36,1,0,JAM2,htxt,0);
486 Np6      /*h0.8----- Font berechnen und schreiben -----
487 HE      -----*/
488 40      BYTE MakeFont(pflag,lochar,hichar)
489 h9      BYTE pflag;
490 a5      short lochar,hichar;
491 dq      ULONG datei;
492 H3      BYTE mode,hmem[100],*nam;
493 OF      short modulo=0,anz=hichar-lochar+1,anzl,h1;
494 D1      USHORT *point=&sublock1[5];
495 yJ      void WriteLine(),WriteCharLoc(),WriteSpace(),WriteKern();
496 Nm      mode=PruConf(edrawer,YMatrix);
497 JM      if(WriteFont(mode,edrawer,YMatrix)<0)return(-1);
498 Ip      sprintf(hmem,"%s/%1.0f",edrawer,(float)YMatrix);
499 tD      Shure(8);
500 RO      for(i=lochar;i<=hichar;i++)
501 Np6      Ascii=i;AsciiOut();
502 HE      fontbreit[i]=Breit(pflag,i);

```

```

415 YO      return();
416 AYC      case MOUSEBUTTONS:
417 SgF      if(code != SELECTDOWN)break;
418 I8      i=Prurec(mox,moy,sdat,2,9);
419 aQ      if(i<0)break;
420 K1      SetAPen(rp,1);
421 nM      switch(1)
422 sAI      case 2:
423 QpL      YMatrix=(long)input_zahl(wn,61,36,2,"
424 q2      ");
425 mv      if(YMatrix<0 || YMatrix>50) YMatrix
426 yHI      =8;
427 JxL      break;
428 3t      case 3:
429 qz      lochar=(short)input_zahl(wn,163,36,3,
430 4OI      "");
431 8ZL      if(lochar>hichar || lochar<0) locha
432 nu      r=32;
433 u3      break;
434 AVI      case 4:
435 OaL      hichar=(short)input_zahl(wn,273,36,3,
436 4N      "");
437 Fp      if(hichar<lochar || hichar>255) hic
438 z8      har=255;
439 HdI      break;
440 1aL      case 5:
441 4M      if(proflag==1)break;
442 Ku      proflag=1;
443 4D      Invert(rp,sdat[5]);Invert(rp,sdat[6])
444 YyH      ;
445 2GL      break;
446 mV      case 6:
447 8PO      if(proflag==0)break;
448 1FR      proflag=0;
449 tWL      Invert(rp,sdat[5]);Invert(rp,sdat[6])
450 Yp      ;
451 kd      break;
452 DM      case 7: case 8:
453 bOH      Invert(rp,sdat[1]);
454 BPI      if(i==7)
455 lz      if(MakeFont(proflag,lochar,hichar)
456 xv      <0)
457 ES      Shure(2);
458 JS      Ascii=help;AsciiOut();
459 U5C      Delay(5);
460 dn      goto ende;
461 100      break;
462 9o      case 9:
463 11      Invert(rp,sdat[i]);
464 Pu      GetFilename(edrawer,efname);
465 JH      YMatrix=(long)atoi(efname);
466 dq3      Invert(rp,sdat[i]);
467 eE      break;
468 3A      RefreshGadgets(&s_gad,wn,NULL);
469 Zg      zahl_out(rp,lochar,hichar);
470 Vm      /*---- Unterfunktion zu h0.7 -----*/
471 kt      void zahl_out(rp,lochar,hichar)
472 rLO      struct RastPort *rp;
473 40      short lochar,hichar;
474 h9      char htxt[10];
475 a5      sprintf(htxt,"%2.0f",(float)YMatrix);
476 dq      Text_out(rp,61,36,1,0,JAM2,htxt,0);
477 H3      Text_out(rp,163,36,1,0,JAM2,htxt,0);
478 OF      Text_out(rp,273,36,1,0,JAM2,htxt,0);
479 D1      Text_out(rp,61,36,1,0,JAM2,htxt,0);
480 yJ      Text_out(rp,163,36,1,0,JAM2,htxt,0);
481 Nm      Text_out(rp,273,36,1,0,JAM2,htxt,0);
482 JM      /*h0.8----- Font berechnen und schreiben -----
483 Ip      -----*/
484 tD      BYTE MakeFont(pflag,lochar,hichar)
485 RO      BYTE pflag;
486 Np6      short lochar,hichar;
487 HE      ULONG datei;
488 40      BYTE mode,hmem[100],*nam;
489 h9      short modulo=0,anz=hichar-lochar+1,anzl,h1;
490 a5      USHORT *point=&sublock1[5];
491 dq      void WriteLine(),WriteCharLoc(),WriteSpace(),WriteKern();
492 H3      mode=PruConf(edrawer,YMatrix);
493 OF      if(WriteFont(mode,edrawer,YMatrix)<0)return(-1);
494 D1      sprintf(hmem,"%s/%1.0f",edrawer,(float)YMatrix);
495 yJ      Shure(8);
496 Nm      for(i=lochar;i<=hichar;i++)
497 JM      Ascii=i;AsciiOut();
498 Ip      fontbreit[i]=Breit(pflag,i);

```



```

488 2m      modulo+=fontbreit[1];
489 pT3      modulo/=8;modulo+=(4-modulo%4);
490 AY      h1=0x6e+modulo*YMatrix;
491 vz      anz1=modulo*YMatrix+anz*8+8;
492 6T      point++;
493 Wg      *point=(anz1+112)/4;point+=4;
494 DB      *point=(anz1+112)/4;point=&ublock1[27];
495 GF      *point=anz1;point++;
496 hk      *point=YMatrix;point++;
497 11      *point=98;
498 CZ      point++;
499 Bh      *point=(USHORT)XMatrix;point++;
500 xP      *point=(USHORT)BaseLine;point+=3;
501 mc      *point=256*lochar+hichar;point+=3;
502 nJ      *point=modulo;point+=2;
503 DY      *point=h1;point+=2;
504 11      *point=h1+anz*4+4;point+=2;
505 Ga      *point=h1+(anz*4+4)+(anz*2+2);
506 qm      point=drawmem;
507 Ga      Shure(8);
508 o1      datei=Open(hmem,MODE_NEWFILE);
509 SY      if(datei==0)return(-1);
510 Mn      Shure(9);
511 Gg      if(Write(datei,ublock1,142)!=142) goto abbruch;
512 Dc      /* ---- CharData schreiben ----*/
513 nd      for(i=0;i<=YMatrix-1;i++)
514 Ez6      BltClear(point,modulo,1);
515 wM      WriteLine(1,lochar,hichar);
516 9m      if(Write(datei,point,modulo)!=modulo) goto abbruch;
517 4L3      /* ---- CharLoc schreiben ----*/
518 j1      WriteCharLoc(lochar,hichar);
519 a0      if(Write(datei,point,anz*4+4)!=anz*4+4) goto abbruch;
520 VC      /* ---- CharSpace schreiben ----*/
521 P2      WriteSpace(lochar,hichar);
522 FX      if(Write(datei,point,anz*2+2)!=anz*2+2) goto abbruch;
523 OJ      /* ---- CharKern schreiben ----*/
524 Te      WriteKern(lochar,hichar);
525 1a      if(Write(datei,point,anz*2+2)!=anz*2+2) goto abbruch;
526 eC      if(Write(datei,ublock2,46)!=46) goto abbruch;
527 VO      Close(datei);
528 ez      Shure(9);
529 JG      /* ---- Fast Direktory erzeugen ----*/
530 Rm      if(mode != 0)
531 h86      sprintf(hmem,"%s/fd.fdir",edrawer);
532 a8      DeleteFile(hmem);
533 UC3      if(mode==2)
534 TY6      nam=AllocMem(1024,MEMF_CLEAR);
535 Jw      if(nam != 0)
536 fD9      strcpy(hmem,edrawer);
537 3X      Parent(hmem);
538 TS      sprintf(hmem,"%s/fd.fdir",hmem);
539 HJ      datei=Open(hmem,MODE_OLDFILE);
540 I5      if(datei != 0)
541 jC0      Close(datei);
542 60      Shure(10);
543 9d      Parent(hmem);
544 Lt      GetDir(nam,hmem,1);
545 9R      Shure(10);
546 Rx9      FreeMem(nam,1024);
547 ID3      return(0);
548 YEO      abbruch:
549 zK3      Shure(9);
550 s1      Close(datei);
551 wX      return(-1);
552 C10      /*--- Unterfunktionen zu h0.8-----*/
553 2x      void WriteLine(1,lochar,hichar)
554 r1      short i,lochar,hichar;
555 0z      register short j,k;
556 oJ      register ULONG *adr;
557 eN      register long count=0;
558 y3      long *point=drawmem;
559 tQ3      for(j=lochar;j<=hichar;j++)
560 G16      adr=cmatrix+100*j+2*i;
561 fK      for(k=0;k<=fontbreit[j]-1;k++,count++)
562 9I9      if(TestBit(adr,k))
563 fJC      SetBit(point,count);
564 v19      else
565 OrC      ClrBit(point,count);
566 MA0      /*-----*/
567 ZM      void WriteCharLoc(lochar,hichar)
568 5a      short lochar,hichar;
569 fp      register short *point=drawmem,offs=0,i;
570 oN3      for(i=lochar;i<=hichar;i++)

```

```

571 kC6      Ascii=1;AsciiOut();
572 x0      *point=offs;point++;
573 Jc      *point=fontbreit[1];point++;
574 Q6      offs+=fontbreit[1];
575 jW3      *point=0;point++;*point=0;
576 Fq0      /*-----*/
577 VM      void WriteKern(lochar,hichar)
578 Fk      short lochar,hichar;
579 YG      register short *point=drawmem,i;
580 yX3      for(i=lochar;i<=hichar;i++)
581 uM6      Ascii=1;AsciiOut();
582 Vu      *point=1;
583 Zw      point++;
584 Tr3      *point=0;
585 Zx0      /*-----*/
586 NV      void WriteSpace(lochar,hichar)
587 Ot      short lochar,hichar;
588 hP      register short *point=drawmem,i;
589 7g3      for(i=lochar;i<=hichar;i++)
590 3V6      Ascii=1;AsciiOut();
591 KP      if(fontbreit[1]==0)
592 1g9      *point=XMatrix;
593 OB6      else
594 nT9      *point=fontbreit[1]+1;
595 186      point++;
596 f33      *point=0;
597 OX0      /*h0.9----- Ermittelt die Breite eines Zeichens
-----*/
598 Bn      short Breit(pflag,ascii)
599 1A      BYTE pflag;
600 JM      short ascii;
601 q4      ULONG *adr=cmatrix+100*ascii;
602 8Z      long i,j,flag=0;
603 zq3      if(pflag==0)return(XMatrix);
604 gZ      for(i=XMatrix-1;i>=0;i--)
605 h96      for(j=0;j<=YMatrix;j++)
606 1C9      if(TestBit(adr,j*64+1))
607 JhC      flag=1;
608 ob3      if(flag==1)break;
609 35      if(flag==0)return(0);
610 ms      return(i+1);
611 tF0      /*h0.10----- Prüft ob Diskfont vorhanden -----
-----*/
612 1M      BYTE PruConf(drawer,high)
613 OJ      char drawer[];
614 fc      short high;
615 s5      ULONG datei;
616 s2      char hmem[100],h;
617 oR      struct FileLock *lock;
618 d73      sprintf(hmem,"%s/%1.0f",drawer,(float)high);
619 Zb      datei=Open(hmem,MODE_OLDFILE);
620 O4      if( datei != 0 ) { Close(datei);return(0); }
621 n0      lock=Lock(drawer,ACCESS_READ);
622 ZV      if(lock != 0) h=1;
623 sf      else
624 Nv6      lock=CreateDir(drawer);
625 9V      h=2;
626 RW3      UnLock(lock);
627 C1      return(h);
628 Dg0      /*h0.11-----Erzeugt den File xxxx.font -----
-----*/
629 Jr      BYTE WriteFont(mode,drawer,high)
630 E7      BYTE mode;
631 I1      char drawer[];
632 xu      short high;
633 mW      char puffer[264],filename[100];
634 tq      long help;
635 Wo      short i;
636 DQ      ULONG datei;
637 MK3      if(mode==0)return(0);
638 5G      if (datei==0) {Shure(11);
639 rxI      return(-1);}
640 RC3      for(i=0;i<=263;i++) puffer[i]=0;
641 xJ      puffer[0]=0x0f;puffer[3]=1;puffer[263]=0x62;puffer[261]
=high;
642 Lf      for(i=strlen(drawer);i>=0;i--)
643 Ms6      if(drawer[i]=='/' ]] drawer[i]=':');
644 JS9      break;
645 xz3      sprintf(filename,"%s.font",drawer);
646 51      sprintf(&puffer[4],"%s/%1.0f",&drawer[i+1],(float)high)
;

```

Listing 2. (Fortsetzung)


```

647 aD      switch(mode)
648 U16      case 1:
649 x39          datei=Open(filename,MODE_OLDFILE);
650 jP          if(datei==0)return(-1);
651 Nw          if(Read(datei,&help,4)!=4) goto abbruch;
652 Gx          help++;
653 Rw          Seek(datei,0,OFFSET_BEGINNING);
654 CL          if(Write(datei,&help,4)!=4) goto abbruch;
655 4n          Seek(datei,0,OFFSET_END);
656 vm          if(Write(datei,&puffer[4],260)!=260)goto abbruch;
657 Wf          break;
658 gy6      case 2:
659 yF9          datei=Open(filename,MODE_NEWFILE);
660 6m          if(datei==0) { shure(11);
661 DJN              return(-1);}
662 vF9          if(Write(datei,puffer,264)!=264) goto abbruch;
663 c1          break;
664 8R3          Shure(11);
665 jC          Close(datei);
666 D8          return(0);
667 T90      abbruch:
668 CV3          Shure(11);
669 ng          Close(datei);
670 rS          return(-1);
671 xD0      /*h0.12----- Ermittelt übergeordnetes Verzeichnis ----
-----*/
672 DS      void Parent(name)
673 Mf      char *name;
674 PT      register short i,flag;
675 KL3          for(i=strlen(name);i>=0;i--)
676 H06              if(name[i]=='/') { flag=0;break; }
677 H7              if(name[i]=='.' ) { flag=1;break; }
678 cF3          if(flag==1)l++;
679 ww          name[i]=0;
680 yP0      /*h0.13----- Zeichnet kleines Image -----
-----*/
681 Qu      void DrawLittle(mode,x,y,adr)
682 4x      BYTE mode;
683 qp      short x,y;
684 sN      register ULONG *adr;
685 4Z      register ULONG *adr1=drawmem;
686 nM      register short i;
687 YI3          for(i=0;i<=99;*adr1=*adr,adr++,adr1++,i++);
688 iB          oima->ImageData=mima->ImageData=drawmem;
689 oe          if(mode==0)
690 1T6              DrawImage(rast,oima,x,y);
691 yI3          else
692 zP6              DrawImage(rast,mima,x,y);
693 x10      /*-----
-----*/

```

(C) 1988 M&T

Listing 2. (Schluß)

Programmname: h1.c

Computer: siehe Listing 1

Sprache: C

Programm : h1.c

```

-----
1 Fo0 /*---(FontDesigner)----- File h1.c -----*/
2 9E #include <exec/memory.h>
3 Rg #include <intuition/intuition.h>
4 L9 #include <intuition/intuitionbase.h>
5 3q #include <libraries/dos.h>
6 i5 #include <graphics/gfxmacros.h>
7 MB #include "header.h"
8 LW /*----- Externe Variablen aus main -----
-----*/
9 hh extern struct IntuitionBase *IntuitionBase;
10 1C extern struct GfxBase *GfxBase;
11 ZC extern ULONG DiskFontBase,DosBase;
12 MH extern BYTE *drawmem;
13 4C extern long *cmatrix,Ascii;
14 VY /*----- Daten für input_zahl() -----
-----*/
15 Bx unsigned char bmaske[]=
16 Cf {
17 tI3      128,64,32,16,8,4,2,1

```

```

18 bX0 };
19 Pa UBYTE rawkey[]={1,2,3,4,5,6,7,8,9,10,57,18,58,68,61,62,63,4
5,46,47,29,30,31,
20 9ZA      15,60,74,67,70,65,69,79,78};
21 on0 UBYTE ascii[]={49,50,51,52,53,54,55,56,57,48,46,101,45,13,5
5,56,57,52,53,54,
22 s59      49,50,51,48,46,45,13,1,8,27,2,3};
23 rD0 /*h1.1----- Zeichnet eine Linie -----
-----*/
24 MB void Line(rp,xa,ya,xe,ye,co,drmd)
25 hx struct RastPort *rp;
26 U4 short xa,ya,xe,ye;
27 KM BYTE co,drmd;
28 Or {
29 KP3      SetAPen(rp,co);SetDrMd(rp,drmd);
30 Ye      Move(rp,xa,ya);
31 Av      Draw(rp,xe,ye);
32 W10 }
33 bI /*h1.2----- Zeichnet ein Rechteck -----
-----*/
34 8k void Box(rp,xa,ya,xe,ye,co,drmd)
35 r7 struct RastPort *rp;
36 eE short xa,ya,xe,ye;
37 UW BYTE co,drmd;
38 Y1 {
39 UZ3      SetAPen(rp,co);SetDrMd(rp,drmd);
40 io      Move(rp,xa,ya);
41 jB      Draw(rp,xe,ya);Draw(rp,xe,ye);
42 yw      Draw(rp,xa,ye);Draw(rp,xa,ya);
43 hC0 }
44 tR /*h1.3----- Installiert ein Fenster -----
-----*/
45 ZT ULONG Make_Window (x,y,w,h,name,flags,iflags,screen)
46 KT short x,y,w,h;
47 GZ char *name;
48 Zq ULONG flags,iflags;
49 LJ struct Screen *screen;
50 kD {
51 oU struct Window *wn;
52 gT struct NewWindow NewWindow;
53 VZ3      NewWindow.LeftEdge=x;
54 bC      NewWindow.TopEdge=y;
55 QA      NewWindow.Width=w;
56 5V      NewWindow.Height=h;
57 Ve      NewWindow.DetailPen=0;
58 SG      NewWindow.BlockPen=1;
59 Gy      NewWindow.Title=name;
60 Mf      NewWindow.Flags=flags;
61 6V      NewWindow.IDCMPFlags=iflags;
62 e7      if(screen==NULL)
63 RY6          NewWindow.Type=WBENCHSCREEN;
64 Gm3      else NewWindow.Type=CUSTOMSCREEN;
65 fS      NewWindow.FirstGadget=NULL;
66 n9      NewWindow.CheckMark=NULL;
67 t7      if (screen==NULL)
68 xm6          NewWindow.Screen=IntuitionBase->FirstScreen;
69 xy3      else NewWindow.Screen=screen;
70 WM      NewWindow.BitMap=NULL;
71 PD      NewWindow.MinWidth=w;
72 wQ      NewWindow.MinHeight=h;
73 hX      NewWindow.MaxWidth=w;
74 Ek      NewWindow.MaxHeight=h;
75 db6          wn=OpenWindow(&NewWindow);
76 yk          return(wn);
77 Fk0 }
78 tJ /*h1.4----- Holt Intuition-Message -----
-----*/
79 KZ BYTE ExaIntui (wn,class,code)
80 Hx struct Window *wn;
81 KG ULONG *class;
82 Mq USHORT *code;
83 Hk {
84 BU struct IntuiMessage *mesg;
85 Am3      mesg=GetMsg(wn->UserPort);
86 RW      if (mesg != NULL)
87 Lo      {
88 U76          *class=mesg->Class;
89 Yz          *code=mesg->Code;
90 cg          ReplyMsg(mesg);
91 Ow          return(1);
92 Uz3      }
93 yt      return(0);
94 W10 }

```



```

95 RJ /*h1.5----- Gibt Text aus -----
-----*/
96 xF void Text_out (rp,x,y,apen,bpen,drmd,text,maxlen)
97 r7 struct RastPort *rp;
98 t1 short x,y,maxlen;
99 nb BYTE apen,bpen,drmd;
100 Og char *text;
101 Z2 {
102 i93 Move(rp,x,y+rp->TxBaseline);
103 2p SetAPen(rp,apen);SetBPen(rp,bpen);SetDrMd(rp,drmd);
104 9n if(maxlen==0)Text(rp,text,strlen(text));
105 gH else Text(rp,text,maxlen);
106 iD0 }
107 pE /*h1.6----- Zeichnet ein ausgefülltes Polygon -----
-----*/
108 uQ BYTE FDrawPolygon(rp,koord,co,drmd)
109 3J struct RastPort *rp;
110 ya short koord[];
111 g1 BYTE co,drmd;
112 kD {
113 V0 struct AreaInfo ainfo;
114 Kq ULONG speicher;
115 Z4 short abuffer[250],i;
116 bg struct TmpRas tmp;
117 Dj3 speicher=drawmem;
118 2V InitArea(&ainfo,abuffer,50);
119 zm rp->TmpRas=(struct TmpRas *)InitTmpRas(&tmp,speicher,20
480);
120 IS rp->AreaInfo=&ainfo;
121 zm SetDrMd(rp,drmd);SetAPen(rp,co);
122 y5 AreaMove(rp,koord[0],koord[1]);
123 5S i=2;
124 yE while(koord[i]>=0)
125 xQ {
126 dx6 AreaDraw(rp,koord[i],koord[i+1]);
127 sF i+=2;
128 4Z3 }
129 fW AreaEnd(rp);
130 Rv rp->TmpRas=NULL;rp->AreaInfo=NULL;
131 ea return(1);
132 8d0 }
133 4p /*h1.7----- Ermittelt Mausklickfeld -----
-----*/
134 yh BYTE Prurec (mox,moy,feld,anf,end)
135 FV register short mox,moy;
136 nG short feld[][4];
137 eO register BYTE anf,end;
138 Ad {
139 27 register BYTE i;
140 wF3 for (i=anf;i<=end;i++)
141 Dg {
142 yf6 if (mox >= feld[i][0] && mox <= feld[i][2] && moy >
= feld[i][1]
143 CHC && moy <= feld[i][3])
144 TJ9 return(i);
145 Lq3 }
146 P0 return(-1);
147 Ns0 }
148 ak /*h1.8----- Eingabe einer double Zahl -----
-----*/
149 Ja double input_zahl(wind,xa,ya,maxlen,defs)
150 6N struct Window *wind;
151 sx short xa,ya;
152 N1 UBYTE maxlen;
153 Xj char *defs;
154 Qt {
155 8s struct IntuiMessage *msg;
156 o4 struct RastPort *rp;
157 Ln ULONG class,iflags;
158 hq UBYTE code,i,pruef(),cpos=0;
159 sh char ps[31];
160 OA double atof();
161 G7 void crs_out();
162 wA iflags=wind->IDCMPFlags;
163 aD ModifyIDCMP(wind,MOUSEBUTTONS[RAWKEY]);
164 vW maxlen--;
165 5f rp=wind->RPort;
166 D3 for (i=0;i<=29;i++)ps[i]=32;
167 l1 ps[30]=0;
168 qx ya=rp->TxBaseline;
169 wu Move(rp,xa,ya);SetDrMd(rp,JAM2);Text(rp,ps,maxlen+1);
170 MN if(strlen(defs) !=0)
171 hA {

```

```

172 ib3 strepy(ps,defs);
173 ke cpos=strlen(defs);
174 vz ps[cpos]=32;
175 pK0 }
176 LW ps[maxlen+1]=0;
177 HX Move(rp,xa,ya);SetDrMd(rp,JAM2);Text(rp,ps,strlen(ps));
178 AR crs_out (rp,cpos,xa,ya);
179 R83 FOREVER
180 qJ {
181 4J9 while(msg=(struct IntuiMessage *)GetMsg(wind->Use
rPort))
182 sL {
183 XyC class=msg->Class;
184 GG code=msg->Code;
185 rz ReplyMsg(msg);
186 ew switch (class)
187 xQ {
188 UsF case MOUSEBUTTONS:
189 mOL if(code != SELECTDOWN)break;
190 qk crs_out(rp,cpos,xa,ya);
191 Sd ModifyIDCMP(wind,iflags);
192 6B return (atof(ps));
193 PhF case RAWKEY:
194 ba code=pruef(code);
195 NM if (code==128)break;
196 lb switch (code)
197 7a {
198 i1I case 1: /* Delete */
199 VML if(cpos==maxlen)
200 Ad {
201 OYO ps[cpos]=32;ps[cpos+1]=0;break;
202 G1L }
203 k8 for(i=cpos;i<=(maxlen-1);i++)ps[i]=ps
[i+1];
204 Y1 ps[maxlen]=32;
205 EN break;
206 X4I case 8: /* Backspace */
207 QOL if(cpos==0)break;
208 Ax cpos--;
209 qE for(i=cpos;i<=(maxlen-1);i++)ps[i]=ps
[i+1];
210 er ps[maxlen]=32;
211 KT break;
212 OEI case 2: /* Cursor left */
213 JEL if(cpos>0)cpos--;
214 NW break;
215 PgI case 3: /* Cursor right */
216 REL if(cpos<maxlen)cpos++;
217 QZ break;
218 xpI case 13: /* Carrige Return */
219 JDL crs_out(rp,cpos,xa,ya);
220 v6 ModifyIDCMP(wind,iflags);
221 Ze return (atof(ps));
222 i1I case 27: /* ESC */
223 PzL for(i=0;i<=maxlen;i++)ps[i]=32;
224 4s cpos=0;
225 Yh break;
226 MGI default:
227 1KL for(i=maxlen;i>=(cpos+1);i--)ps[i]=ps
[i-1];
228 lB ps[cpos]=code;
229 CN ps[maxlen+1]=0;
230 9e if (cpos<maxlen)cpos++;
231 JEI }
232 ou Move(rp,xa,ya);
233 8N SetDrMd(rp,JAM2);
234 JT Text(rp,ps,strlen(ps));
235 ZT crs_out(rp,cpos,xa,ya);
236 oJC }
237 pK9 }
238 qL6 }
239 rM0 }
240 rJ /*----- Unterfunktionen zu h1.8 -----*/
241 i7 void crs_out(rp,cpos,xa,ya)
242 CS struct RastPort *rp;
243 Pu UBYTE cpos;
244 NS short xa,ya;
245 tM {
246 lq register BYTE i;
247 cJ3 Move(rp,xa+TextLength(rp," ",1)*cpos,ya);

```

Listing 3. Geben Sie »h1.c« bitte mit dem Checksummer ein


```

248 4Q SetDrMd(rp,COMPLEMENT] INVERSVID);
249 nk Text(rp," ",1);
250 2X0 }
251 8v UBYTE pruef(code)
252 w9 USHORT code;
253 1U {
254 uC register UBYTE i;
255 ur3 for(i=0;i<=31;i++)
256 4X {
257 676 if(code==rawkey[i]) return(ascii[i]);
258 Af3 }
259 kb return(128);
260 Ch0 }
261 S4 /*h1.9----- Öffnet Libraries -----
-----*/
262 dL BYTE Oplibs()
263 Be {
264 In3 IntuitionBase=OpenLibrary("intuition.library",0L);
265 XH GfxBase=OpenLibrary("graphics.library",0L);
266 4Q DosBase=OpenLibrary("dos.library",0L);
267 72 DiskfontBase=OpenLibrary("diskfont.library",0L);
268 J1 if(DiskfontBase==0)
269 Hk {
270 FV6 Shure(4);
271 eD CloseLibrary(IntuitionBase);
272 oK CloseLibrary(GfxBase);
273 qE exit(0);
274 Qv3 }
275 up return(0);
276 Sx0 }
277 hL /*h1.10----- Schließt Libraries -----
-----*/
278 uO void Cllibs()
279 Ru {
280 PD3 CloseLibrary(DosBase);
281 oK CloseLibrary(DiskfontBase);
282 yU CloseLibrary(GfxBase);
283 qP CloseLibrary(IntuitionBase);
284 a50 }
285 es /*h1.11----- Invertier BS-Bereich -----
-----*/
286 24 void Invert(rp,dat)
287 vB struct RastPort *rp;
288 tg short dat[];
289 b4 {
290 KF3 SetDrMd(rp,COMPLEMENT);
291 O2 RectFill(rp,dat[0]+1,dat[1]+1,dat[2]-1,dat[3]-1);
292 ID0 }
293 V1 /*h1.12----- Prüft ob Bit gesetzt -----
-----*/
294 N7 BYTE TestBit(adR,bioff)
295 zY register UBYTE *adr;
296 Cs register long bioff;
297 jC {
298 Bn3 adr+=bioff>>3;
299 AX if((*adr & bmaske[bioff%8]) != 0)
300 NJ6 return(1);
301 KF3 return(0);
302 sN0 }
303 qm /*h1.13----- Bestimmtes Bit setzen -----
-----*/
304 kC void SetBit(adR,bioff)
305 91 register UBYTE *adr;
306 M2 register long bioff;
307 tM {
308 Lx3 adr+=bioff>>3;
309 8P *adr|=bmaske[bioff%8];
310 OV0 }
311 gY /*h1.14----- Bestimmtes Bit löschen -----
-----*/
312 ZG void ClrBit(adR,bioff)
313 Hq register UBYTE *adr;
314 UA register long bioff;
315 1U {
316 T53 adr+=bioff>>3;
317 bU *adr&=~bmaske[bioff%8];
318 8d0 }
319 4M /*h1.15----- Zeichensatz verschieben -----
-----*/
320 19 short mdat[][4]=
321 7a {
322 oh3 { 10,16,128,28 }, { 138,16,262,28 }, { 86,33,188,45 },
323 ea { 10,50,64,62 }, { 208,50,262,62 },

```

```

324 XT0 };
325 7t char *mtxt[]=
326 Cf {
327 NZ3 "FROM ASCII:", "UNTIL ASCII:", "TO ASCII:", " OK", "CANCEL"
,
328 bX0 };
329 Y5 short moveas[3]={ 65,128,128 };
330 Hz void MoveAscii()
331 Hk {
332 L1 struct Window *wn;
333 fv struct RastPort *rp;
334 XN ULONG flags=WINDOWDRAG] ACTIVATE;
335 AU ULONG class;
336 IV USHORT code;
337 xD short i,mox,moy,*point,offs;
338 xC void moveout(),MAusf();
339 9c3 wn=Make_Window(120,100,280,71,"MOVE",flags,MOUSEBUTTONS,
0);
340 1P if(wn==0)return();
341 WG rp=wn->RPort;
342 SG SetAPen(rp,1);RectFill(rp,0,11,280,71);
343 30 Box(rp,2,12,277,69,0,JAM1);
344 41 for(i=0;i<=4;i++)
345 Vy {
346 5V6 SetAPen(rp,0);
347 1v RectFill(rp,mdat[i][0],mdat[i][1],mdat[i][2],mdat[i][
3]);
348 1r Box(rp,mdat[i][0],mdat[i][1],mdat[i][2],mdat[i][3],2,
JAM1);
349 Ne Box(rp,mdat[i][0]-1,mdat[i][1],mdat[i][2]+1,mdat[i][3
],2,JAM1);
350 QS Text_out(rp,mdat[i][0]+3,mdat[i][1]+3,3,0,JAM2,mtxt[i
],0);
351 fA3 }
352 1X moveout(rp);
353 Fw FOREVER
354 e7 {
355 ke6 while(ExaIntui(wn,&class,&code))
356 g9 {
357 zS9 if(code!=SELECTDOWN)break;
358 XD i=Prurec(wn->MouseX,wn->MouseY,mdat,0,4);
359 oS if(i<0)break;
360 oN switch(i)
361 1E {
362 n1C case 0: case 1: case 2:
363 6NF point=&moveas[i];
364 ce offs=mdat[i][0]+strlen(mtxt[i])*8+3;
365 U2 *point=(short)input_zahl(wn,offs,mdat[i][1]+
3,3,"");
366 JQ if(*point<0)*point=0;
367 AT if(*point>255)*point=255;
368 rO break;
369 3MC case 3:
370 d1F Invert(rp,mdat[i]);
371 Qa MAusf();
372 8SC case 4:
373 goF Invert(rp,mdat[i]);
374 Kb Delay(5);
375 dx CloseWindow(wn);
376 v1 return();
377 5a9 }
378 Rx6 moveout(rp);
379 7c }
380 8d3 }
381 9e0 }
382 yT /*---- Unterfunktion zu h1.15 ----*/
383 O1 void moveout(rp)
384 Uk struct RastPort *rp;
385 9c {
386 uZ short offs,i;
387 H5 char text[5];
388 gL3 for(i=0;i<=2;i++)
389 Dg {
390 Yj6 sprintf(text,"%3.0f", (float)moveas[i]);
391 35 offs=mdat[i][0]+strlen(mtxt[i])*8+3;
392 oB Text_out(rp,offs,mdat[i][1]+3,3,0,JAM2,text,0);
393 Lq3 }
394 Mr0 }
395 Bg /*---- Unterfunktion zu h1.15 ----*/
396 sX void MAusf()
397 Lo {
398 ZD short mofa=moveas[0],moua=moveas[1],mota=moveas[2],help;

```



```

399 FA register short j,i,step=1;
400 6p long *adr,*adr1;
401 Kn long merker=Ascii;
402 Yf3 if(mofa==moua)return();
403 Zp if(mofa>moua) { help=mofa;mofa=moua;moua=help; }
404 K2 if(mota > mofa && mota < moua)
405 Tw {
406 aJ6 mota+=(moua-mofa);
407 YA help=mofa;mofa=moua;moua=help-1;
408 1A step=-1;
409 b63 }
410 Cp else moua++;
411 xC for(i=mofa;i!=moua;i+=step,mota+=step)
412 a3 {
413 5R6 if(mota>255)continue;
414 oY adr=cmatrix+100*i;
415 Sm adr1=cmatrix+100*mota;
416 Ww for(j=0;j<=99;j++,adr1++)*adr1=adr[j];
417 In Ascii=i;
418 XQ AsciiOut();
419 lG3 }
420 3e Ascii=merker;
421 aT AsciiOut();
422 oJ0 }
423 bP /*-----
-----*/

```

(C) 1987 M&T

Listing 3. (Schluß)

Programmname: h2.c
 Computer: siehe Listing 1
 Sprache: C

Programm : h2.c

```

1 RMO /*--(FontDesigner)----- File h2.c -----*/
2 9E #include <exec/memory.h>
3 Rg #include <intuition/intuition.h>
4 L9 #include <intuition/intuitionbase.h>
5 3q #include <libraries/dos.h>
6 15 #include <graphics/gfxmacros.h>
7 MB #include "header.h"
8 OB /*----- Externe Variablen aus main -----
-----*/
9 Ip extern long XMatrix,YMatrix,BaseLine,HelpLine,Ascii,DMode;
10 9R extern long Quadrant;
11 OU extern struct TextFont *tftopaz,*tftuser;
12 FJ extern struct Window *wind;
13 ek extern struct RastPort *rast;
14 Vi extern ULONG *cmatrix,matrix[],cmemo[],scmemo[];
15 5d extern UBYTE bmaske[],drawmem;
16 nB extern struct Image *oima,*drima,*mima;
17 EJ /*----- Daten für Hauptdisplay -----
-----*/
18 ls short reedat[][4]=
19 F1 {
20 oH3 { 0,11,512,211 },
21 Ua { 456,216,505,228 }, /* Change Ascii */
22 Eh { 456,228,505,239 },
23 18 { 456,239,505,250 },
24 m6 { 520,187,566,199 }, /* 4 */
25 2y { 566,187,597,199 },
26 9z { 597,187,628,199 },
27 us { 520,204,566,216 }, /* 7 */
28 4b { 566,204,597,216 },
29 Be { 597,204,628,216 },
30 Bu { 520,221,566,233 }, /* 10 */
31 vQ { 566,221,597,233 },
32 2R { 597,221,628,233 },
33 s1 { 520,238,566,250 }, /* 13 */
34 W8 { 566,238,597,250 },
35 49 { 597,238,628,250 },
36 Wa { 594,13,628,27 }, /* 16 */

```

```

37 CF { 594,35,628,49 }, /* 17 */
38 zg { 594,57,628,71 }, /* 18 */
39 fL { 594,79,628,93 }, /* 19 */
40 7d { 594,101,628,115 }, /* 20 */
41 lr { 520,145,556,163 }, /* 21 */
42 72 { 592,145,628,163 },
43 Bm { 556,127,592,145 },
44 gF { 556,163,592,181 }, /* 24 */
45 Cn { 556,145,574,154 },
46 9u { 574,145,592,154 },
47 Cn { 556,154,574,163 },
48 tg { 574,154,592,163 }, /* 28 */
49 5J { 10,216,64,228 }, /* MIRR_B */
50 HE { 10,238,64,250 }, /* MIRR_H */
51 JC { 79,216,133,228 }, /* ROTATE */
52 Bc { 302,216,356,228 }, /* UNDO */
53 2s { 79,238,133,250 }, /* INVERS */
54 UB { 233,216,287,228 }, /* LOAD */
55 Mr { 148,216,218,228 }, /* EXPAND_X */
56 Pu { 148,238,218,250 }, /* EXPAND_Y */
57 9T { 233,238,287,250 }, /* SAVE */
58 UA { 371,216,441,228 }, /* PRG */
59 bK { 371,238,441,250 }, /* SHOW */
60 JG { 302,238,356,250 }, /* MOVE */
61 IEO };
62 As char *rectxt[]=
63 xQ {
64 kS3 " ", " ", " ", " ",
65 r0 "XM=", " ", " ", "YM=", " ", " ", "BL=", " ", " ", "HL=", " ", " ",
66 ap "ALL", "MAT", "DEL", "SNP", "CPY",
67 nV " ", " ", " ", " ",
68 oW " ", " ", " ", " ",
69 lo "MIRR_H", "MIRR_B", "ROTATE", "UNDO", "INVERS", "LOAD", "EXPAN
D_X", "EXPAND_Y",
70 ZX "SAVE", "PROGRAMM", "SHOW", "MOVE",
71 S00 };
72 4s char txtotf[][2]=
73 7a {
74 te3 { 0,0 }, { 11,3 }, { 0,0 }, { 0,0 },
75 PM { 3,3 }, { 0,0 }, { 0,0 },
76 QN { 3,3 }, { 0,0 }, { 0,0 },
77 RO { 3,3 }, { 0,0 }, { 0,0 },
78 SP { 3,3 }, { 0,0 }, { 0,0 },
79 HB { 5,4 }, { 5,4 }, { 5,4 }, { 5,4 }, { 5,4 }, /* 16-20 */
80 yz { 0,0 }, { 0,0 }, { 0,0 }, { 0,0 }, /* 21-24 */
81 ZO { 0,0 }, { 0,0 }, { 0,0 }, { 0,0 }, /* 25-28 */
82 L7 { 3,3 }, { 3,3 }, { 3,3 }, { 11,3 },
83 9x { 3,3 }, { 11,3 }, { 3,3 }, { 3,3 },
84 Yu { 11,3 }, { 3,3 }, { 19,3 }, { 11,3 },
85 gc0 };
86 0a short pfeil[][7]=
87 Lo {
88 zn3 { 528,154,545,146,545,162,-2 },
89 d8 { 620,154,603,146,603,162,-2 },
90 I4 { 574,129,560,143,588,143,-2 },
91 9y { 574,179,560,165,588,165,-2 },
92 Ap { 576,193,586,188,586,198,-2 },
93 us { 617,193,607,188,607,198,-2 },
94 4s { 581,206,573,214,589,214,-2 },
95 jP { 612,214,604,206,620,206,-2 },
96 mm { 581,223,573,231,589,231,-2 },
97 RJ { 612,231,604,223,620,223,-2 },
98 oA { 576,244,586,239,586,249,-2 },
99 YD { 617,244,607,239,607,249,-2 },
100 pc { 480,229,472,237,488,237,-2 },
101 DM { 480,249,472,241,488,241,-2 },
102 xt0 };
103 ZI char wtitle[]="F O N T DESIGNER Programmed by Richard Ar
etz";
104 aZ /*h2.1----- Erzeugt das Hauptdisplay -----
-----*/
105 OX BYTE Display()
106 e7 {
107 3h ULONG flags=WINDOWCLOSE|ACTIVATE|REPORTMOUSE|SMART_REFRESH|
108 NrC RMBTRAP|WINDOWDEPTH;
109 KIO ULONG iflags=CLOSEWINDOW|MOUSEBUTTONS|VANILLAKEY;
110 3L short i;
111 OG3 wind=Make_Window(0,0,636,256,wtitle,flags,iflags,0);

```

Listing 4. »h2.c« bitte mit dem Checksummer eingeben.
 Beachten Sie die Anweisungen im Text und in Tabelle 2a
 beziehungsweise 2b (Seite 54).

```

112 MO    if(wind==0) return(-1);
113 Tb    rast=wind->RPort;
114 Sv    SetAPen(rast,3);RectFill(rast,0,11,640,256);
115 Xt    SetAPen(rast,0);
116 U9    RectFill(rast,0,11,512,211);
117 b5    RectFill(rast,521,14,587,65);
118 5t    RectFill(rast,521,70,587,121);
119 AX    for(i=21;i<=24;i++)
120 sL    {
121 rk6        RectFill(rast,recdat[i][0]+4,recdat[i][1]+2,
122 QCN            recdat[i][2]+4,recdat[i][3]+2);
123 zU3    }
124 pW    Box(rast,0,11,512,211,2,JAM1);
125 qo    for(i=1;i<=40;i++)
126 yR    {
127 Jj6        if ( !(i>20 && i<29) )
128 OT        {
129 l79            SetAPen(rast,0);
130 Ot            RectFill(rast,recdat[i][0]+4,recdat[i][1]+2,
131 ZLN            recdat[i][2]+4,recdat[i][3]+2);
132 8d6        }
133 uH        SetAPen(rast,1);
134 4k        RectFill(rast,recdat[i][0],recdat[i][1],recdat[i][2],
135 4g            recdat[i][3]);
136 rW        Box(rast,recdat[i][0],recdat[i][1],recdat[i][2],recda
t[i][3],2,JAM1);
137 KVF        Text_out(rast,recdat[i][0]+txtoff[i][0],recdat[i][1]+
txtoff[i][1],0,1,
138 E]3            JAM2,rectxt[i],0);
139 IZ        for (i=0;i<=13;i++)
140 t46            FDrawPolygon(rast,pfeil[i],2,JAM1);
141 b03        AsciiOut();Matout();Mem_to_Matrix();
142 fh        Invert(rast,recdat[17-DMode]);
143 RN        Invert(rast,recdat[25+Quadrant]);
144 n1        return(0);
145 Lq0 }
146 G6 /*h2.2----- Aktuellen ascii-code ausgeben -----
-----*/
147 5T void AsciiOut()
148 Kn {
149 RF    char text[5];
150 U13    sprintf(text,"%3.0f",(float)Ascii);
151 T5    Text_out(rast,recdat[1][0]+txtoff[1][0],recdat[1][1]+txt
off[1][1],
152 CMC        0,1,JAM2,text,0);
153 Ty0 }
154 ZT /*h2.3----- Werte für Matrix ausgeben -----
-----*/
155 a4 void Matout()
156 Sv {
157 Mq    char text[10],i;
158 DA    long help;
159 pR3    for(i=4;i<=13;i+=3)
160 Wz    {
161 bA6        switch(i)
162 Y1        {
163 6j9            case 4: help=XMatrix;break;
164 Iz            case 7: help=YMatrix;break;
165 ye            case 10: help=BaseLine;break;
166 Dg            case 13: help=HelpLine;break;
167 hc6        }
168 J4        sprintf(text,"%s%2.0f",rectxt[i],(float)help);
169 J1        Text_out(rast,recdat[1][0]+txtoff[1][0],recdat[1][1]+
txtoff[1][1],
170 UeF            0,1,JAM2,text,0);
171 lG3    }
172 mH0 }
173 cI /*h2.4----- Aktuellen Ascii-code verändern -----
-----*/
174 Xy void ChangeAscii(mode)
175 tm    BYTE mode;
176 mF    {
177 cw        ULONG class;
178 kx        USHORT code;
179 cB        register short i;
180 bR3        if(mode==0)
181 rK        {
182 cy6            SetAPen(rast,0);
183 GO            Ascii=(long)input_zahl(wind,recdat[1][0]+txtoff[1][0]
,
184 rxT            recdat[1][1]+txtoff[1][1],3,""
);

```

```

185 oy6        if(Ascii<0 || Ascii>255)Ascii=65;
186 ng        AsciiOut();
187 6K        Mem_to_Matrix();
188 tj        return();
189 3Y3    }
190 RL        Invert(rast,recdat[mode+1]);
191 n5        Delay(10);
192 eL        FOREVER
193 3W        {
194 nX6            switch (mode)
195 5Y            {
196 CT9                case 1:
197 jPC                    if(Ascii<255)Ascii++;
198 pC                    else Ascii=0;
199 8H                    break;
200 Ia9                case 2:
201 HCC                    if(Ascii>0)Ascii--;
202 uo                    else Ascii=255;
203 CL                    break;
204 In6            }
205 6z            AsciiOut();
206 uk            DrawLittle(0,0,0,cmatrix+100*Ascii);
207 Xm            Delay(3);
208 z4            while(ExaIntui(wind,&class,&code))
209 Jm            {
210 y99                switch(class)
211 Lo                {
212 sGC                    case MOUSEBUTTONS:
213 WkF                        Mem_to_Matrix();
214 pj                        Invert(rast,recdat[mode+1]);
215 KA                        return();
216 Uz9                }
217 V06            }
218 W13        }
219 X20 }
220 L4 /*h2.5----- Matrixdaten verändern -----
-----*/
221 4n void ChangeMatrix(var)
222 AG    BYTE var;
223 XO    {
224 Nh        ULONG class;
225 Vi        USHORT code;
226 ZF        long *point,pmax,mode=(var-1)%3;
227 Cb3        Mabox(0);
228 ga        switch (var)
229 d6        {
230 JQ6            case 4: case 5: case 6:
231 aW9                point=&XMatrix;pmax=64;break;
232 p56            case 7: case 8: case 9:
233 SK9                point=&YMatrix;pmax=50;break;
234 AL6            case 10: case 11: case 12:
235 ud9                point=&BaseLine;pmax=50;break;
236 jo6            case 13: case 14: case 15:
237 dE9                point=&HelpLine;pmax=64;break;
238 qL3        }
239 YO        if(mode==0)
240 oH        {
241 Zv6            SetAPen(rast,0);
242 NT            *point=(long)input_zahl(wind,recdat[var][0]+26,recdat
[var][1]+3,2,"");
243 SW            if(*point<1)*point=1;
244 eW            if(*point>pmax)*point=pmax;
245 QP            Matout();
246 3H            Mem_to_Matrix();
247 qg            return();
248 OV3        }
249 da        Invert(rast,recdat[var]);
250 k2        Delay(10);
251 bI        FOREVER
252 OT        {
253 c16            Mabox(0);
254 lV            switch (mode)
255 3W            {
256 AR9                case 1:
257 hrC                    if(*point>0)*point-=1;
258 dJ                    else *point=pmax;
259 6F                    break;
260 GY9                case 2:
261 uQC                    if(*point<pmax)*point+=1;
262 AM                    else *point=1;
263 Fk6            }
264 qG            Mabox(1);
265 kJ            Matout();

```



```

266 Rf      Delay(2);
267 Va      while (ExaIntui(wind,&class,&code))
268 Gj      {
269 zH9      switch (class)
270 I1      {
271 pDC      case MOUSEBUTTONS:
272 OxF      Invert(rast,recdat[var]);
273 U1      Mem_to_Matrix();
274 H7      return();
275 Rw9      }
276 Sx6      }
277 Ty3      }
278 Uz0      }
279 7Z      /*h2.6----- Zeichnet die Matrixbox -----
-----*/
280 a0      void Mabox(mode)
281 bU      BYTE mode;
282 Ux      {
283 f3      short xe=XMatrix*8,ye=11+YMatrix*4,yb=11+BaseLine*4,xh=Help
Line*8;
284 on3      Box(rast,0,11,xe,ye,mode+2,JAM1);
285 p5      Line(rast,0,yb,xe,yb,mode+2,JAM1);
286 M0      Line(rast,xh,11,xh,ye,mode+2,JAM1);
287 d80      }
288 H6      /*h2.7-----Setzt einen Punkt in die Matrix -----
-----*/
289 1R      void SetMPPoint(sp,ze,mode)
290 Nt      BYTE sp,ze,mode;
291 d6      {
292 gb      short xa=sp*8+1,ya=11+ze*4+1,spmax=63,zemax=49;
293 3V      ULONG *adr=cmatrix+100*Ascii+2*ze;
294 qW3      if(DMode==0) { spmax=XMatrix-1;zemax=YMatrix-1; }
295 fQ      if(sp>spmax || sp<0)return();
296 st      if(ze<0 || ze>zemax)return();
297 zI      SetAPen(rast,mode);SetDrMd(rast,JAM1);
298 R3      RectFill(rast,xa,ya,xa+6,ya+2);
299 xX      WritePixel(rast,522+sp,15+ze);
300 ga      if(sp>=32){ adr++;sp-=32; }
301 eQ      if(mode==1)
302 1r6      *adr*=maske[sp];
303 1V3      else
304 xz6      *adr&=**maske[sp];
305 vQ0      }
306 7P      /*h2.8----- Hauptfunktion in Matrix zeichnen -----
-----*/
307 qA      void DrawInto(mode)
308 2v      BYTE mode;
309 v0      {
310 15      ULONG class;
311 t6      USHORT code;
312 LZ      short mox,moy;
313 x6      ULONG iflags=wind->IDCMPFlags;
314 nm      void SetMPPoint();
315 pq3      ModifyIDCMP(wind,wind->IDCMPFlags|=MOUSEMOVE);
316 DD      SetMPPoint(wind->MouseX/8,(wind->MouseY-11)/4,mode);
317 fM      FOREVER
318 4X      {
319 LQ6      while (ExaIntui(wind,&class,&code))
320 6Z      {
321 p79      switch (class)
322 8b      {
323 f3C      case MOUSEBUTTONS:
324 bmE      ModifyIDCMP(wind,iflags);
325 6w      return();
326 2fC      case MOUSEMOVE:
327 eYF      mox=wind->MouseX;moy=wind->MouseY;
328 wP      SetMPPoint(mox/8,(moy-11)/4,mode);
329 Jo9      }
330 Kp6      }
331 Lq3      }
332 Mr0      }
333 25      /*h2.9----- Zeichen aus Zeichensatz in Zwischenspeiche
r -----*/
334 ky      void Font_to_Mem(ascii)
335 hy      long ascii;
336 Mp      {
337 ao      ULONG *adr=cmatrix+100*ascii;
338 nq      UBYTE *fadr=tfuser->tf_CharData;
339 e7      USHORT *loc=tfuser->tf_CharLoc;
340 gX      register long hohe,breite,offset,btloff;
341 MR      register long i,j;
342 Qq      BYTE TestBit();
343 6r      void SetBit(),AsciiOut();

```

```

344 N23      if(ascii<=tfuser->tf_HiChar && ascii>=tfuser->tf_LoC
har)
345 Vy      {
346 1r6      for(i=0;i<=99;i++)adr[i]=0;
347 5R      offset=ascii-tfuser->tf_LoChar;
348 tB      if(offset<0)offset=0;
349 Ju      hohe=tfuser->tf_YSize-1;
350 N3      btloff=loc[offset*2];
351 Jf      breite=loc[offset*2+1]-1;
352 aR      if(hohe<0 || hohe>51) hohe=8;
353 X1      if(breite<0 || breite>64)return();
354 iO      for(i=0;i<=hohe;i++)
355 f8      {
356 1J9      for(j=0;j<=breite;j++)
357 hA      {
358 2LC      if(TestBit(fadr,j+btloff))
359 RsF      SetBit(adr,j);
360 oJ9      }
361 F1      fadr+=tfuser->tf_Modulo;adr+=2;
362 qL6      }
363 rM3      }
364 n3      Ascii=ascii;
365 gZ      AsciiOut();
366 uP0      }
367 sQ      /*h2.10----- Zeichen Manipulieren -----
-----*/
368 1C      void Delete(mode)
369 1u      BYTE mode;
370 uN      {
371 ru      ULONG *adr=cmatrix+100*Ascii,*point=cmemo;
372 rw      register long i,j;
373 An3      switch(mode)
374 yR      {
375 qB6      case 0: /* CLR */
376 L99      switch(DMode)
377 1U      {
378 6MC      case 0:
379 mZF      for(i=0;i<=(YMatrix-1);i++,adr+=2)
380 4X      {
381 bMI      for(j=0;j<=(XMatrix-1);j++)
382 6Z      {
383 WmL      ClrBit(adr,j);
384 ChI      }
385 DiF      }
386 9IC      break;
387 HY      case 1:
388 RXF      for(i=0;i<=99;i++)adr[i]=0;
389 Hm9      }
390 DM6      break;
391 5S      case 1: /* Snapshot */
392 bP9      switch(DMode)
393 Hk      {
394 McC      case 0:
395 MIF      for(i=0;i<=(YMatrix-1);i++,adr+=2,point+=2)
396 Kn      {
397 rcI      for(j=0;j<=(XMatrix-1);j++)
398 Mp      {
399 UcL      if(TestBit(adr,j))
400 Qk0      SetBit(point,j);
401 I5L      else
402 9IO      ClrBit(point,j);
403 VOI      }
404 WfF      }
405 SbC      break;
406 ar      case 1:
407 LcF      for(i=0;i<=99;i++,point++)*point=adr[i];
408 VeC      break;
409 b69      }
410 XD      DrawLittle(1,0,0,cmemo);
411 Yh6      break;
412 Cs      case 2: /* Copy */
413 wk9      switch(DMode)
414 c5      {
415 hxC      case 0:
416 hDF      for(i=0;i<=(YMatrix-1);i++,adr+=2,point+=2)
417 f8      {
418 CxI      for(j=0;j<=(XMatrix-1);j++)
419 hA      {

```

Listing 4. (Fortsetzung)

```

420 B9L          if(TestBit(point,j))
421 Rs0           SetBit(adr,j);
422 dQL          else
423 AQ0           ClrBit(adr,j);
424 qLI          }
425 rMF          }
426 nWC          break;
427 vC           case 1:
428 VCF           for(i=0;i<=99;i++,adr++)*adr=point[i];
429 qzC          break;
430 wR9          }
431 s16          break;
432 yT3          }
433 4I           Mem_to_Matrix();
434 OV0          }
435 eW           /*h2.11----- Scrolling ausführen -----
-----*/
436 9Q           void Scroll(mode)
437 70           BYTE mode;
438 OT           {
439 oe           ULONG *adr=cmatrix+100*Ascii,*point;
440 OG           long i,j,bitoff;
441 g6           BYTE astep,bstep,cstep;
442 fD           long xstart,xend,ystart,yend;
443 uE           ULONG class;
444 2F           USHORT code;
445 7m3          switch(Quadrant)
446 8b           {
447 DT6           case 0:
448 3y9           xstart=0;ystart=0;xend=HelpLine;yend=BaseLine;
449 AJ           break;
450 IZ6           case 1:
451 wJ9           xstart=HelpLine;ystart=0;xend=XMatrix;yend=BaseLin
e;
452 DM           break;
453 Nf6           case 2:
454 tC9           xstart=0;ystart=BaseLine;xend=HelpLine;yend=YMatri
x;
455 GP           break;
456 S16           case 3:
457 gn9           xstart=HelpLine;ystart=BaseLine;xend=XMatrix;yend=
YMatrix;
458 JS           break;
459 Pu3          }
460 5p           switch (mode)
461 Nq           {
462 S16           case 0:
463 NM9           astep=1;bstep=0;cstep=0;
464 PY           break;
465 Xo6           case 1:
466 sO9           astep=-1;bstep=0;cstep=0;
467 Sb           break;
468 cu6           case 2:
469 ZU9           astep=0;bstep=0;cstep=64;yend--;
470 Ve           break;
471 hO6           case 3:
472 cq9           astep=0;bstep=64;cstep=0;yend--;
473 Yh           break;
474 e93          }
475 2a          /*--- Scroll left & right up & down ---*/
476 cb6          adr=cmatrix+100*Ascii;
477 gS           for(i=0;i<=99;i++,adr++)scmemo[i]=*adr;
478 OW           adr=cmatrix+100*Ascii+2*ystart;
479 Md           point=scmemo+2*ystart;
480 I6           for(i=ystart;i<=(yend-1);i++)
481 hA           {
482 QA9           for(j=xstart;j<=(xend-1);j++)
483 Jc           {
484 8vC           if(TestBit(point,j+astep+cstep))
485 KEF           SetBit(adr,j+bstep);
486 rNC           else ClrBit(adr,j+bstep);
487 rM9           }
488 XH           switch (mode)
489 pI           {
490 RFC           case 0: ClrBit(adr,xend);break;
491 8n           case 1: ClrBit(adr,xstart);break;
492 wR9           }
493 3w           adr+=2;point+=2;
494 yT6           }
495 eO           switch (mode)
496 wP           {
497 5N9           case 2:
498 LgC           adr=cmatrix+100*Ascii+2*yend;

```

```

499 UG           for(i=xstart;i<=(xend-1);i++)
500 MbF           ClrBit(adr,i);
501 O9C           break;
502 CV9           case 3:
503 PvC           adr=cmatrix+100*Ascii+2*ystart;
504 ZL           for(i=xstart;i<=(xend-1);i++)
505 RgF           ClrBit(adr,i);
506 Af6           }
507 GU           Mem_to_Matrix();
508 Ch0           }
509 2Y           /*h2.12----- Zeichen aus Zwischenspeicher in Matrix --
-----*/
510 Ex           void Mem_to_Matrix()
511 Be           {
512 AO           register long *adr;
513 8D           register long i,j;
514 NE           register UBYTE *point=drawmem;
515 FE3           adr=cmatrix+100*Ascii;
516 zr           BltClear(drawmem,12800,1);
517 yX           for(i=0;i<=49;i++,point+=256)
518 I1           {
519 RH6           for(j=0;j<=31;j++)
520 Kn           {
521 Gq9           if(*adr & maske[j])
522 Mp           {
523 gfC           *(point+64+j)=127;
524 rJ           *(point+128+j)=127;
525 MG           *(point+192+j)=127;
526 Uz9           }
527 VO6           }
528 IO           adr++;
529 Id           for(j=32;j<=63;j++)
530 Ux           {
531 HD9           if(*adr & maske[j-32])
532 Wz           {
533 qpC           *(point+64+j)=127;
534 1T           *(point+128+j)=127;
535 WQ           *(point+192+j)=127;
536 e99           }
537 fA6           }
538 BY           adr++;
539 hC3           }
540 OK           point=drawmem;
541 rn           drima->ImageData=point;
542 1S           DrawImage(rast,drima,0,0);
543 LB           DrawLittle(0,0,0,cmatrix+100*Ascii);
544 Mm           Mabox(1);
545 nIO           }
546 pF           /*-----*/

```

(C) 1987 M&T

Listing 4. (Schluß)

Programmname: h3.c

Computer: siehe Listing 1

Programm : h3.c

```

1 dy0 /*---(FontDesigner)----- File h3.c -----*/
2 9E #include <exec/memory.h>
3 Rg #include <intuition/intuition.h>
4 L9 #include <intuition/intuitionbase.h>
5 3q #include <libraries/dos.h>
6 15 #include <graphics/gfxmacros.h>
7 MB #include "header.h"
8 q1 /*----- Externe Variablen -----
-----*/
9 g4 extern struct Image *oima,*drima,*mima;
10 kY extern struct FileInfoBlock *f_info;
11 ci extern struct RastPort *rast;
12 FJ extern struct Window *wind;
13 Mt extern long XMatrix,YMatrix,BaseLine,HelpLine,Ascii,DMode;
14 GJ extern long Quadrant,undoascii;
15 D2 extern ULONG *cmatrix;

```



```

16 QL extern BYTE *drawmem;
17 T1 extern ULONG cmemo[],scmemo[],undomem[];
18 hg /*----- Daten für Programmmodus -----*/
19 3U short pdat[4]=
20 GJ {
21 kW3 { 10,16,88,28 }, { 105,16,167,28 }, { 184,16,254,28 },
22 K6 { 10,33,56,45 }, { 76,33,122,45 }, { 142,33,188,45 }, { 208
,33,254,45 },
23 wD { 10,50,56,62 }, { 76,50,122,62 }, { 142,50,164,62 }, { 172
,50,194,62 },
24 by { 202,50,224,62 }, { 232,50,254,61 },
25 mf { 30,67,50,77 }, { 30,87,50,97 }, { 10,77,30,87 }, { 50,77,
70,87 },
26 Y8 { 80,76,118,88 }, { 128,76,166,88 },
27 24 { 176,76,254,88 },
28 lh0 };
29 UJ char *ptxt[] =
30 Qt {
31 Q33 "START:", "END:", "STEP:", " INV", " ROT", "MIR_H", "MIR_B", "E
XP_X", "EXP_Y",
32 DB "Q1", "Q2", "Q3", "Q4", "UP", "DOWN", "LEFT", "RIGHT", "DELE", "
GO", "PZ:",
33 qm0 };
34 WM short parr[7]=
35 Vy {
36 tT3 { 40,68,47,75,33,75,-2 },
37 bM { 40,96,47,89,33,89,-2 },
38 lZ { 13,82,25,79,25,86,-2 },
39 x1 { 67,82,55,79,55,86,-2 },
40 xt0 };
41 lh char programm[100];
42 md /*h3.1----- Zeichen spiegeln -----*/
43 dr void Mirror(mode)
44 mf BYTE mode;
45 f8 {
46 57 register long i,j,*adr=cmatrix+100*Ascii,*point;
47 2b register long bitoff;
48 vY3 switch(mode)
49 jC {
50 o46 case 0:
51 sp9 for(i=0;i<=YMatrix;i++,adr+=2)
52 mF {
53 sOC for(j=0;j<=HelpLine;j++)
54 oH {
55 w4F if(TestBit(adr,j))
56 92I SetBit(adr,2*HelpLine-j-1);
57 vQC }
58 wR9 }
59 s1 break;
60 OH6 case 1:
61 Hn9 bitoff=2*64*(BaseLine-1)+64;point=adr;
62 zF for(i=0;i<=BaseLine;i++,adr+=2,bitoff-=64)
63 xQ {
64 vMC for(j=0;j<=XMatrix;j++)
65 zS {
66 7FF if(TestBit(adr,j))
67 3dI SetBit(point,bitoff+j);
68 6bC }
69 7c9 }
70 3C break;
71 9e3 }
72 FT Mem_to_Matrix();
73 Bg0 }
74 I5 /*h3.2----- Undospeicher in Hauptspeicher -----*/
75 JM void Undo()
76 Ad {
77 zb register long i,*adr=cmatrix+100*Ascii;
78 4c3 for(i=0;i<=99;i++,adr++)*adr=undomem[i];
79 6z Ascii=undoascii;
80 5y AsciiOut();
81 Oc Mem_to_Matrix();
82 Kp0 }
83 BZ /*h3.3----- Zeichen drehen -----*/
84 Zx void Rotate()
85 Jm {
86 N1 register long *adr=cmatrix+100*Ascii,*point=scmemo;
87 zK register long i,j,bitoff;
88 CT3 for(i=0;i<=99;i++,point++)*point=adr[i];
89 BO point=scmemo;

```

```

90 cw for(i=0;i<=XMatrix;i++,point+=2)
91 Ps {
92 WK6 bitoff=(YMatrix-1)*64+i;
93 zB for(j=0;j<=YMatrix;j++,bitoff-=64)
94 Sv {
95 wu9 if(TestBit(point,j))
96 aaC SetBit(adr,bitoff);
97 7j9 else ClrBit(adr,bitoff);
98 a56 }
99 b63 }
100 hv Mem_to_Matrix();
101 d80 }
102 BZ /*h3.4----- Zeichen invertieren -----*/
103 6m void Revers()
104 c5 {
105 ey register long i,j,*adr=cmatrix+100*Ascii,xend=XMatrix,yend=
YMatrix;
106 Nj3 for(i=0;i<=(yend-1);i++,adr+=2)
107 f8 {
108 Sv6 for (j=0;j<=(xend-1);j++)
109 hA {
110 px9 if(TestBit(adr,j))
111 80C ClrBit(adr,j);
112 dQ9 else
113 TuC SetBit(adr,j);
114 qL6 }
115 rM3 }
116 xB Mem_to_Matrix();
117 t00 }
118 Ay /*h3.5----- Zeichen in x-Richtung vergrößern -----*/
119 Yt void Expand_x()
120 sL {
121 z2- register ULONG *adr=cmatrix+100*Ascii,*point=scmemo;
122 pu register long i,j;
123 123 for(i=0;i<=99;i++,point++)*point=adr[i];
124 kx point=scmemo;
125 ie for(i=0;i<=YMatrix-2;i++,adr+=2,point+=2)
126 yR {
127 kX6 for(j=0;j<=XMatrix-1;j++)
128 OT {
129 US9 if(TestBit(point,j))
130 2V {
131 40C SetBit(adr,j*2);SetBit(adr,j*2+1);
132 8d9 }
133 y1 else
134 6Z {
135 LJc ClrBit(adr,j*2);ClrBit(adr,j*2+1);
136 Ch9 }
137 Di6 }
138 Ej3 }
139 KY Mem_to_Matrix();
140 GLO }
141 k6 /*h3.6----- Zeichen in y-Richtung vergrößern -----*/
142 zL void Expand_y()
143 F1 {
144 OQ ULONG *adr=cmatrix+100*Ascii,*point=scmemo;
145 M3 register long i;
146 8P3 for(i=0;i<=99;i++,point++)*point=adr[i];
147 7K point=scmemo;
148 I5 for(i=0;i<=99;i+=4)
149 Lo {
150 aP6 adr[i]=*point;adr[i+2]=*point;point++;
151 ak adr[i+1]=*point;adr[i+3]=*point;point++;
152 Sx3 }
153 Ym Mem_to_Matrix();
154 Uz0 }
155 lW /*h3.7----- Programmmodus aktivieren -----*/
156 t8 void Programm()
157 Tw {
158 qZ static short start=32,end=255,step=1,count=0;
159 Uk ULONG flags=WINDOWCLOSE|ACTIVATE|WINDOWDRAG;
160 KA ULONG iflags=CLOSEWINDOW|MOUSEBUTTONS;
161 aG struct Window *wn;
162 uA struct RastPort *rp;
163 OI ULONG class;
164 WJ USHORT code;

```

Listing 5. Geben Sie »h3.c« bitte mit dem Checksummer (Seite 159) ein

```

165 RC short i,mox,moy,flag=0;
166 US char htext[10];
167 nf void pz_out(),Go_Amiga();
168 3e3 wn=Make_Window(170,140,278,105,"Programm",flags,iflags,0
);
169 xe if(wn==0)return();
170 lV rp=wn->RPort;
171 jw SetAPen(rp,1);RectFill(rp,0,11,278,105);
172 aC Box(rp,2,12,275,103,0,JAM1);
173 Kh for (i=0;i<=19;i++)
174 kD {
175 Kk6 SetAPen(rp,0);
176 pw RectFill(rp,pdat[i][0],pdat[i][1],pdat[i][2],pdat[i][
3]);
177 sw Box(rp,pdat[i][0],pdat[i][1],pdat[i][2],pdat[i][3],2,
JAM2);
178 Nc Box(rp,pdat[i][0]-1,pdat[i][1],pdat[i][2]+1,pdat[i][3
],2,JAM2);
179 nO if(i<13 || i>16)
180 AL8 Text_out(rp,pdat[i][0]+3,pdat[i][1]+3,3,0,JAM2,ptxt
[i],0);
181 KX6 else
182 L69 FDrawPolygon(rp,parr[i-13],3,JAM1);
183 xS3 }
184 20 pz_out(rp,start,end,step);
185 KJ if(count==0)
186 Wb6 Text_out(rp,203,79,3,0,JAM2,"NULL",0);
187 qd3 else
188 lt6 Text_out(rp,203,79,3,0,JAM2,ptxt[programm[count-1]],0
);
189 bI3 FOREVER
190 OT {
191 606 while(ExaIntui(wn,&class,&code))
192 2V {
193 8s9 mox=wn->MouseX;moy=wn->MouseY;
194 m4 switch (class)
195 5Y {
196 amC case CLOSEWINDOW:
197 l5F CloseWindow(wn);
198 3t return();
199 f3C case MOUSEBUTTONS:
200 alF if(code != SELECTDOWN) break;
201 5l i=Prurec(mox,moy,pdat,0,19);
202 5v if(i<0)break;
203 OG if(i>2 && i<18)
204 Eh {
205 DKl if(flag==0)count=0;
206 5G Invert(rp,pdat[i]);
207 du Delay(5);
208 Ha programm[count]=i;
209 WD programm[count+1]=0;
210 OT sprintf(htext,"%s",ptxt[i]);
211 3k Text_out(rp,203,79,3,0,JAM2,htext,5);
212 wI count++;
213 jh if(count>98)count--;
214 yM flag=1;
215 TyF }
216 2T SetAPen(rp,1);
217 xn switch (i)
218 Sv {
219 XnI case 0:
220 jsL start=(short)input_zahl(wn,61,19,3,"")
;
221 q6 if(start<0 || start>255)start=32;
222 Ve break;
223 duI case 1:
224 VzL end=(short)input_zahl(wn,140,19,3,"");
225 Gm if(end<start || end>255)end=255;
226 Zi break;
227 jII case 2:
228 KFL step=(short)input_zahl(wn,227,19,3,"")
;
229 zN if(step<1 || step>255)step=1;
230 dm break;
231 LOI case 18:
232 LzL Text_out(rp,pdat[i][0]+3,pdat[i][1]+3,
1,0,JAM2,
"STOP",0);
233 6LU Invert(rp,pdat[i]);
234 XlL Go_Amiga(wn,start,end,step);
235 YE break;
236 js case 19:
237 UAI Text_out(rp,203,79,3,0,JAM2," NULL",0)
;
238 LML

```

```

239 cn Invert(rp,pdat[i]);
240 UI count=0;programm[0]=0;
241 ox break;
242 uPF }
243 Ta if(i>2)Invert(rp,pdat[i]);
244 Yr Text_out(rp,pdat[18][0]+3,pdat[18][1]+3,3,0,
JAM2,
" GO ",0);
245 Le0 pz_out(rp,start,end,step);
246 20F }
247 zU9 }
248 OV6 }
249 lW3 }
250 2X0 }
251 dT /*---- Unterfunktion zu h3.7 ----*/
252 lI void pz_out(rp,st,end,stp)
253 Nd struct RastPort *rp;
254 Tm short st,end,stp;
255 3W {
256 wu char htext[10];
257 Ok3 sprintf(htext,"%3.0f",(float)st);
258 W9 Text_out(rp,61,19,3,0,JAM2,htext,0);
259 MO sprintf(htext,"%3.0f",(float)end);
260 2C Text_out(rp,140,19,3,0,JAM2,htext,0);
261 WY sprintf(htext,"%3.0f",(float)stp);
262 eL Text_out(rp,227,19,3,0,JAM2,htext,0);
263 FkO {
264 qg /*---- Unterfunktion zu h3.7 ----*/
265 J5 void Go_Amiga(wn,st,end,stp)
266 Hx struct Window *wn;
267 gz short st,end,stp;
268 GJ {
269 oG short i,count;
270 mQ long merl=Ascii,mer2=Quadrant;
271 8S ULONG class;
272 GT USHORT code;
273 NO3 for(i=st;i<=end;i+=stp)
274 Mp {
275 QJ6 Ascii=i;count=0;AsciiOut();
276 UH ExaIntui(wn,&class,&code);
277 OS if(class==MOUSEBUTTONS && code==SELECTDOWN)break;
278 lF while (programm[count] != 0)
279 Ru {
280 cp9 Invert(wn->RPort,pdat[programm[count]]);
281 5x switch (programm[count])
282 Ux {
283 rNC case 3: Revers();break;
284 cD case 4: Rotate();break;
285 tO case 5: Mirror(0);break;
286 yV case 6: Mirror(1);break;
287 ue case 7: Expand_x();break;
288 zl case 8: Expand_y();break;
289 7H case 9: case 10: case 11: case 12:
290 bqF Quadrant=programm[count]-9;
291 cl break;
292 qpC case 13: Scroll(2);break;
293 xy case 14: Scroll(3);break;
294 sr case 15: Scroll(0);break;
295 zO case 16: Scroll(1);break;
296 ez case 17: Delete(0);break;
297 nI9 }
298 uH Invert(wn->RPort,pdat[programm[count]]);
299 l7 count++;
300 qL6 }
301 rM3 }
302 jn Ascii=merl;Quadrant=mer2;
303 eq Mem_to_Matrix();AsciiOut();
304 uPO }
305 gR /*h3.8----- Gesamten Zeichensatz anzeigen -----
-----*/
306 2G void Show()
307 tM {
308 j3 ULONG class;
309 r4 USHORT code;
310 9n short Font_out(),ascii=0;
311 Yx3 Mabox(0);
312 aj ascii=Font_out(ascii);
313 bI FOREVER
314 OT {
315 in6 while(ExaIntui(wind,&class,&code))
316 2V {
317 yY9 switch (code)
318 4X {
319 azC case MENUDOWN:
320 N3F if(ascii>=254)ascii=0;
321 js ascii=Font_out(ascii);

```



```

322 7G      break;
323 GpC      case SELECTDOWN:
324 JXF      Mem_to_Matrix();
325 pF      Mabox(1);
326 7x      return();
327 Hm9      }
328 In6      }
329 Jo3      }
330 Kp0      }
331 8G      /*----- Unterfunktion zu h3.8 -----*/
332 HG      short Font_out(ascii)
333 O3      short ascii;
334 Kn      {
335 73      register long xa=4,ya=13;
336 Jm3      SetAPen(rast,0);SetDrMd(rast,JAM1);
337 4d      RectFill(rast,1,11,511,210);
338 GS      while( ya<160 && ascii<254)
339 Ps      {
340 256      xa=4;
341 nL      while(xa<447 && ascii<254)
342 Sv      {
343 k39      DrawLittle(0,xa-522,ya-15,cmatrix+100*ascii);
344 JF      Box(rast,xa-1,ya-1,xa+XMatrix+1,ya+YMatrix+1,2,JAM
1);
345 aJ      ascii++;
346 RO      xa+=XMatrix+2;
347 b66      }
348 b7      ya+=YMatrix+2;
349 d83      }
350 l1      return(ascii);
351 fAO      }
352 kb      /*h3.9----- Speicher besorgen -----
-----*/
353 t4      BYTE GetMemory(mode)
354 mf      BYTE mode;
355 f8      {
356 Aw      register BYTE *bpoi;
357 if      register short i,j;
358 Yp3      if(mode==0) goto cleanup1;
359 Me      oima=AllocMem(sizeof(struct Image),MEMF_CHIP|MEMF_CLEAR)
;
360 ps      if(oima==0)return(-1);
361 1F      drima=AllocMem(sizeof(struct Image),MEMF_CHIP|MEMF_CLEAR
);
362 SH      if(drima==0)goto cleanup6;
363 Mc      mima=AllocMem(sizeof(struct Image),MEMF_CHIP|MEMF_CLEAR)
;
364 8r      if(mima==0)goto cleanup5;
365 9L      f_info=AllocMem(sizeof(struct FileInfoBlock),MEMF_CLEAR)
;
366 dD      if(f_info==0)goto cleanup4;
367 IL      cmatrix=AllocMem(102400,MEMF_CLEAR);
368 00      if(cmatrix==0) goto cleanup3;
369 Jn      drawmem=AllocRaster(512,400);
370 yG      if(drawmem==0) goto cleanup2;
371 7s      oima->LeftEdge=522;oima->TopEdge=15;oima->Width=64;o
ima->Height=50;
372 aU      oima->Depth=1;oima->ImageData=NULL;oima->PlanePick=1;
373 Gu      oima->PlaneOnOff=0;oima->NextImage=NULL;
374 d1      drima->LeftEdge=0;drima->TopEdge=11;drima->Width=512;
drima->Height=200;
375 P2      drima->Depth=2;drima->ImageData=NULL;drima->PlanePick
=3;
376 tM      drima->PlaneOnOff=0;drima->NextImage=NULL;
377 rO      mima->LeftEdge=522;mima->TopEdge=71;mima->Width=64;m
ima->Height=50;
378 MK      mima->Depth=1;mima->ImageData=NULL;mima->PlanePick=1;
379 6q      mima->PlaneOnOff=0;mima->NextImage=NULL;
380 H6      bpoi=drawmem+12800;
381 2F      for (i=0;i<=199;i++)
382 6Z      {
383 KQ6      for(j=0;j<=63;j++,bpoi++)
384 8b      {
385 nY9      *bpoi=128;
386 fC      if(i%4 == 0)
387 2JC      *bpoi=0xff;
388 Cl6      }
389 Hm3      }
390 lg      return(0);
391 cDO      cleanup1: FreeRaster(drawmem,512,400);
392 hP      cleanup2: FreeMem(cmatrix,102400);
393 QB      cleanup3: FreeMem(f_info,sizeof(struct FileInfoBlock));
394 R3      cleanup4: FreeMem(mima,sizeof(struct Image));

```

```

395 Wp      cleanup5: FreeMem(drima,sizeof(struct Image));
396 1R      cleanup6: FreeMem(oima,sizeof(struct Image));
397 S33      return(-1);
398 Qv0      }
399 dJ      /*h3.11----- Sicherheitsabfragen un Meldungen ausgeben ---
-----*/
400 1J      char *shtxt[]=
401 Ps      {
402 IX3      "Are you shure to quit ?",
403 PW      "I can't find this font !",
404 mp      "ERROR !! Font not saved.",
405 IO      "Not enough Memory !!!",
406 Co      "I can't find Diskfont Library",
407 A1      "No ASSIGN ! Please use CLI",
408 53      "Loading Font ! Please Wait",
409 EM      "Loading Topaz/8 ! Please Wait",
410 q8      "Examine Font !!",
411 WM      "One Moment ! Writing Char Data !",
412 24      "Creat New Fastdirectory !",
413 50      "Writing xxxx.font !",
414 zv0      };
415 qp      BYTE Shure(nr)
416 83      BYTE nr;
417 f8      {
418 WS      static flag=0;
419 GZ      static struct Window *swind;
420 C3      ULONG flags=ACTIVATE|WINDOWDRAG;
421 Ys      ULONG class;
422 gt      USHORT code;
423 DJ      char text[2][4];
424 fD      short sh_data[2][4];
425 YH      short wbr;
426 JF      BYTE i;
427 g23      if(nr>=6 && flag==1)
428 qJ      {
429 Pm6      flag=0;
430 se      CloseWindow(swind);
431 QL      return(0);
432 yT3      }
433 hN      wbr=strlen(shtxt[nr])*8+28;
434 JG      sh_data[0][0]=10;sh_data[0][1]=32;sh_data[0][2]=40;sh_da
ta[0][3]=44;
435 9k      if(nr>0) sh_data[0][2]=32;
436 ef      sh_data[1][0]=wbr-44;sh_data[1][1]=32;
437 ft      sh_data[1][2]=wbr-22;sh_data[1][3]=44;
438 r8      if(nr<1)
439 1U      {
440 vO6      strcpy(text[0],"YES");strcpy(text[1],"NO");
441 7c3      }
442 xk      else
443 5Y      {
444 FS6      strcpy(text[0],"OK");strcpy(text[1],text[0]);
445 Bg3      }
446 AP      swind=Make_Window(0,0,wbr,51,"REQUEST",flags,MOUSEBUTTON
S,0);
447 13      if(swind==0)return(0);
448 TM      SetAPen(swind->RPort,1);RectFill(swind->RPort,0,11,wbr
,51);
449 DA      Box(swind->RPort,2,12,wbr-3,49,0,JAM1);
450 KO      Text_out(swind->RPort,10,16,0,1,JAM2,shtxt[nr],0);
451 Jf      if(nr<6)
452 Eh      {
453 gK6      for(i=0;i<=1;i++)
454 GJ      {
455 6I9      Box(swind->RPort,sh_data[i][0],sh_data[i][1],sh_d
ata[i][2],
456 hbN      sh_data[i][3],2,JAM1);
457 8a9      Text_out(swind->RPort,sh_data[i][0]+3,sh_data[i][
1]+3,0,1,JAM2,
458 icI      text[i],0);
459 Pu6      }
460 Qv3      }
461 G3      else
462 Or      {
463 zN6      flag=1;
464 xs      return(0);
465 VO3      }
466 41      FOREVER
467 Tw      {
468 Ep6      while(ExaIntui(swind,&class,&code))
469 Vy      {
470 oH9      if(code!=SELECTDOWN)break;
471 Ju      i=Prurec(swind->MouseX,swind->MouseY,sh_data,0,1
);

```

Listing 5. (Fortsetzung)

```

472 RH      if(i<0)break;
473 ZL      CloseWindow(swind);
474 nd      return(1);
475 fA6     }
476 gB3     }
477 hC0 }
478 JO /*h3.12----- Direktory sortieren -----
-----*/
479 qT void SortDir(names,anz)
480 62 BYTE *names;
481 oV short anz;
482 1B {
483 Qk register short i,j=1;
484 Ip BYTE *p1=names,*p2,*p3,h;
485 U93 p3=AllocMem(1024,MEMF_CLEAR);
486 nB if(p3==0)return();
487 Lt p2=p3;
488 jE for(i=0;i<=1023;i++,p1++)p2[i]=p1;
489 6K p1=names;
490 W6 for(i=1;i<=anz-1;i++)
491 rK {
492 E46 h=*p2;p2++;
493 Am if(h==1)
494 uN {
495 vv9 *p1=1;p1++;
496 Ln strcpy(p1,p2);
497 8p p1+=strlen(p2)+1;
498 2X6 }
499 Dv p2+=strlen(p2)+1;
500 4Z3 }
501 Z7 p2=p3;
502 iI for(i=1;i<=anz-1;i++)
503 3W {
504 QG6 h=*p2;p2++;
505 Ju if(h==0)
506 6Z {
507 159 *p1=0;p1++;
508 Xz strcpy(p1,p2);
509 K1 p1+=strlen(p2)+1;
510 EJ6 }

```

```

511 P7      p2+=strlen(p2)+1;
512 GL3     }
513 pw      FreeMem(p3,1024);
514 In0 }
515 SL /*h3.13----- Zeichensatzdiskette aktualisieren --
-----*/
516 yS void Assign(drive)
517 Ro char *drive;
518 1I {
519 eC struct FileLock *lockp,*Lock();
520 OT char text[100],*point,i;
521 sm3 strcpy(text,drive);text[4]=0;
522 vk if((lockp=Lock(text,ACCESS_READ))==0)
523 I86 return();
524 3H3 if(Examine(lockp,f_info)==0)
525 KA6 return();
526 He3 UnLock(lockp);
527 f1 /*-- Stern und Anführungszeichen im Filenamen bearbeiten
--*/
528 1V point=f_info->fib_FileName;
529 FO strcpy(text,f_info->fib_FileName);
530 T8 for(i=0;i <= strlen(text);i++,point++)
531 Vy {
532 jw6 *point=text[i];
533 zt if(text[i]!='*' ] ] text[i]=' ')
534 Y1 {
535 oc9 *point='*';point++;
536 n0 *point=text[i];
537 fA6 }
538 gB3 }
539 sg /*-----*/
540 3k sprintf(text,"assign > nil: fonts: \"%s:%s\"",
541 byB f_info->fib_FileName,drive+4);
542 7d3 Execute(text,0,0);
543 lG0 }
544 YM /*-----*/
-----*/

```

(C) 1987 M&T

Listing 5. (Schluß)

COUPON

Name

Telefon

Adresse

Abo für Zeitschrift/
ab Ausgabe

Datum/Unterschrift

Zahlungsweise 1/4 1/2 1/1

Unterschrift des Erziehungs-
berechtigten bei Jugend-
lichen unter 18 Jahren

Super-Software Abonnement:

Wußten Sie, daß Sie die Disketten zu den Listings aus den Markt&Technik-Zeitschriften im Abonnement bestellen können? Für nur **DM 298,-** jährlich erhalten Sie jeden Monat die Diskette aus einer dieser Zeitschriften: 64'er, Amiga-Magazin, PC Magazin Plus oder ST-Magazin.

Für Schüler und Studenten gibt's Ermäßigung: **DM 278,-** jährlich (bitte Kopie des Ausweises beifügen). Sie können vierteljährlich, halbjährlich oder jährlich bezahlen.

Sie sparen sich durch ein Abo mehr als **DM 50,-** und die Zeit für die Bestellabwicklung - deshalb: am besten gleich den Coupon ausschneiden und ausgefüllt an die genannte Adresse schicken!

Das Abo kann innerhalb von 8 Tagen widerrufen werden.


Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Markt&Technik Verlag AG
Unternehmensbereich Buchverlag
Hans-Pinsel-Str. 2, 8013 Haar bei München



Amiga kunterbunt

Die Grafik-Bibliothek des Amiga stellt umfangreiche Funktionen zur Verfügung, einzelne Bildteile zu kopieren und sie gleichzeitig zu verfremden. Allerdings kommt manchmal der Wunsch auf, dieses auch auf einzelne Farben zu beschränken. Nach einer ergebnislosen Suche in der »graphics.library« ist der C-Programmierer dann auf sich selbst angewiesen. Steht man mit dem Blitter eventuell auf Kriegsfuß, dann sieht das Ganze ziemlich hoffnungslos aus.

Die hier vorgestellten Funktionen des Programms »BlitColor« (Listing 1) sollen nun helfend eingreifen und die Grundlage für das »farbweise« Bearbeiten von Bildern schaffen. Dies ist einigen Anwendern zum Beispiel durch die Programme »PixMate« oder »Butcher« bekannt.

Was steckt hinter der Maske?

Um die Funktionen verstehen zu können, muß zunächst einmal der Begriff »Maske« näher erläutert werden. Eine Maske ist, wenn man so will, eine Art Tabelle, in der vermerkt wird, ob an einer bestimmten Position ein Zustand auftritt oder nicht. Auf eine spezielle Farbe einer Grafik bezogen gibt die Maske also einen Hinweis, ob diese Farbe vorhanden ist oder nicht. Möchte man beispielsweise für einen 320 x 200 großen und vier Bitplanes tiefen Screen eine Maske erstellen, so bleibt nichts anderes übrig, als bei jedem Punkt »nachzuschauen«, ob er die entsprechende Farbe hat oder nicht. Dieses wird dann mit »1« oder »0« in die Tabelle eingetragen. Demnach muß die Maske auch genau die Größe unserer Farbgrafik haben, also 320 x 200 Bit. Es ist dabei auch egal, wie »tief« unsere Grafik ist, denn die Maske enthält ja nur die Information über das Auftreten einer einzigen Farbe. Soll nur ein Ausschnitt aus dem Screen benutzt werden, so kann die Maske entsprechend kleiner sein. Letztendlich ist eine Maske mit einer Grafik gleichzusetzen, die aber immer nur eine Bitplane benötigt.

Daß der Amiga mit Farben nicht gerade geizt, ist allgemein bekannt. Aber wie steht es um Funktionen zur Bearbeitung und Manipulation bestimmter Farben in einer Grafik? Wir haben genau auf diesem Bereich ein Defizit entdeckt, das mit »BlitColor« behoben wird.

Um nun eine Maske dieser Art zu erstellen, wird die Funktion »GetColorMask« aufgerufen. Sie brauchen nur die notwendigen Parameter zu übergeben, die folgendermaßen definiert sind:

```
void GetColorMask (SourceBitMap, SourceX, SourceY,
MaskBitMap, MaskX, MaskY,
SizeX, SizeY, Color)
struct BitMap *Source
BitMap, *MaskBitMap;
WORD MaskX, MaskY, SizeX,
SizeY; UBYTE Color;
```

»SourceBitMap« gibt die Bitmap der Grafik an, in der Sie Veränderungen vornehmen wollen. Diese muß natürlich entsprechend ermittelt oder initialisiert werden.

»MaskBitMap« ist der Grafikbereich, den Sie als Maske benutzen wollen. Auch diese Bitmap-Struktur sollte richtig initialisiert sein. »SourceX« und »SourceY« legen fest, ab welcher X- und Y-Position mit der Maskierung begonnen werden soll.

»MaskX« und »MaskY« bewirken, daß die maskierten Daten auch innerhalb der Maskbitplane verschiedene Positionen annehmen können. Die Größe des Bereichs wird mit »SizeX« und »SizeY« eingestellt, und die Farbe wird letztendlich mit »Color« ausgewählt.

Eine Farbe wird ersetzt...

Haben Sie erst einmal die Maske für einen bestimmten Bereich in der Grafik ermittelt, so läßt sich diese nun dazu verwenden, die festgelegten Punkte durch eine andere Farbe zu ersetzen. Dazu rufen Sie die Funktion »CreateColor« auf. Im Vergleich zu »GetColor-

Mask« sind nur die beiden Bitmap-Parameter vertauscht, da ja nun der umgekehrte Vorgang stattfindet.

Mit anderen Worten: Ist in der Maskbitplane ein Pixel gesetzt, so bekommt der entsprechende Pixel in der Destinationbitplane — ehemals Sourcebitplane — die mit Color ausgewählte Farbe; ist der Maskbitplane-Pixel gelöscht, so bleibt der Pixel in der Destinationbitmap unverändert. Alle weiteren Parameter sind identisch. Dieser Vorgang läßt sich mit der Routine »ReplaceColor« auch auf einmal erledigen; es müssen nur zwei Farben angegeben werden, nämlich die zu ersetzende und die neue Farbe.

...oder ausgetauscht

Möchten Sie nun zwei Farben austauschen, so ist es umgänglicher, sich eine zweite MaskBitMap anzulegen, da nun das Auftreten von zwei Farben vermerkt werden muß. Daher benötigt die Funktion »ExchangeColor« drei weitere Argumente. Dies ist der Zeiger auf eine weitere Bitmap-Struktur und die Werte der X- und Y-Koordinate in der zweiten Maskbitplane. Die beiden auszutauschenden Farben werden notwendigerweise auch angegeben.

Wenn sich der Guru meldet

Innerhalb der Funktionen erfolgt keine Prüfung der übergebenen Argumente. Die Funktionen sind daher recht empfindlich gegenüber falsch übergebenen Parametern und rächen sich meist mit einem verwüsteten Speicher. Um ganz sicher zu gehen, sollten Sie die Maskbitplane immer auf die Größe der zu bearbeitenden Bitplanes bringen, so

daß wirklich jeder Bereich manipuliert werden kann. Wer mit dem Gedanken spielt, sich seine Maskbitplanes selbst anzulegen, der sollte bedenken, daß diese im Chip-Memory liegen müssen. Unsere Routinen greifen nämlich auf die Funktion »BlitBitMap« der »graphics.library« zurück, die den Blitter benötigt. Dieser wiederum gehört zu den Custom-chips, die nur die untersten 512K des Speichers adressieren können, also das sogenannte Chip-Memory.

BitMap ist nicht gleich Rastport

Sie werden sich bestimmt schon gewundert haben, warum bisher nur Bitmaps zur Festlegung der Grafik benutzt wurden und nicht die dafür sonst häufig benutzten Rastports? Nun, das liegt einfach daran, daß Rastports die Bearbeitung von überlappenden Speicherbereichen innerhalb der Bitplanes ermöglichen, das sogenannte »Clipping«. Dies tritt zum Beispiel auf, wenn Sie in der Workbench ein Fenster auf ein anderes legen und diese sich somit teilweise oder ganz überlagern. Zur Verwaltung der daraus resultierenden »Schnittspeicherbereiche« wird die »layer.library« benutzt, was natürlich zusätzlichen Rechenaufwand erfordert und Grafikoperationen bremst. Unsere Routinen arbeiten mit Bitmaps und sind daher sehr schnell.

Möchten Sie allerdings als Programmierer die Möglichkeit des »Clippings« nicht missen, so wird einfach in unserem C-Quelltext die Funktion »BlitBitMap« durch »ClipBlit« ausgetauscht. Diese führt im Prinzip die gleiche Operation durch, jedoch auf Rastports bezogen. Ebenso muß vorher die Angabe der zu verarbeitenden Bitplanes — das zehnte Argument beim BlitBitMap-Aufruf — durch den Makro »SetWrMsk()« ersetzt werden, da sich hierfür extra ein Element in der Rastport-Struktur befindet. Als letzter Schritt wird überall »BitMap« zu »RastPort« umbenannt; es soll ja mit RastPorts gearbeitet werden.

Unser Demo-Programm (Listing 2) generiert zunächst eine Zufallsgrafik in verschiede-

nen Farben. Mittels der Exchange-Funktion werden dann auf der linken Screenhälfte Farben vertauscht. Das Programm endet mit einer neuen Variante, ein Bild auszu-
blenden: Die Farben werden mit »ReplaceColor« nachein-

ander zusammengerafft. Der Einfachheit halber werden hier für die beiden MaskBitPlanes zwei Screens geöffnet, die Sie sich während der Ausführung des Programms einmal anschauen sollten.

(Thomi Mauch/
Christian Wolffrs)

Programmname:	BlitColor.h
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztek-C V3.4 und Lattice-C V4.0
Aufrufe:	abhängig vom verwendeten Compiler
Bemerkung:	Die Routine ist in eigene C-Programme mit »#include« einzubinden

Programm : BlitColor.h

```

1 R50 /*****
2 Cq /**/
3 Z7 /**/ Routinen zum Bearbeiten von Farben /**/
4 Es /**/
5 kp /**/   written in 1988 by Thomi Mauch /**/
6 Gu /**/
7 XB /*****
8 iU struct BitMap TmpBitMap;
9 qN InitTmpBitMap (BitMap)
10 2W struct BitMap *BitMap;
11 7a {
12 zX3 /* interner Gebrauch: verbiegt Pointer einer BitMap */
13 3h WORD i;
14 TG TmpBitMap.BytesPerRow = BitMap->BytesPerRow;
15 SC TmpBitMap.Rows = BitMap->Rows;
16 B2 TmpBitMap.Flags = BitMap->Flags;
17 2x TmpBitMap.pad = BitMap->pad;
18 uR TmpBitMap.Depth = 8;
19 Br for (i=0; i<8; i++)
20 TG6 TmpBitMap.Planes[i] = BitMap->Planes[0];
21 Lq0 }
22 7k GetColorMask (SrcBitMap,SrcX,SrcY,MaskBitMap,MaskX,MaskY,SizeX,SizeY,Color)
23 Kx struct BitMap *SrcBitMap,*MaskBitMap;
24 Im WORD SrcX,SrcY,MaskX,MaskY,SizeX,SizeY;
25 d1 BYTE Color;
26 Mp {
27 YE3 /**/ extrahiert eine Farbe aus einer BitMap und /**/
28 Vn /**/ stellt sie auf einer einzigen BitPlane als /**/
29 UT /**/ Maske dar /**/
30 Ky WORD i;
31 xa UBYTE Minterm;
32 Q1 InitTmpBitMap (MaskBitMap);
33 G3 if (Color & 1)
34 A76 Minterm = 0xc0; /* Dest = Source */
35 OB3 else
36 ZV6 Minterm = 0x30; /* Dest = NOT Source */
37 zs3 BltBitMap (SrcBitMap,SrcX,SrcY,&TmpBitMap,MaskX,MaskY,SizeX,SizeY,Minterm,1,NULL);
38 N2 for (i=1; i<SrcBitMap->Depth; i++)
39 Z26 {
40 tm if (Color & (1<<i))
41 E89 Minterm = 0x80; /* Dest = Source AND Dest */
42 VI6 else
43 Tc9 Minterm = 0x20; /* Dest = NOT Source AND Dest */
44 h6 BltBitMap (SrcBitMap,SrcX,SrcY,&TmpBitMap,MaskX,MaskY,SizeX,SizeY,Minterm,1<<i,NULL);
45 jE }
46 kFO }
47 mm CreateColor (MaskBitMap,MaskX,MaskY,DestBitMap,DestX,DestY,SizeX,SizeY,Color)
48 5V struct BitMap *MaskBitMap,*DestBitMap;
49 JG WORD MaskX,MaskY,DestX,DestY,SizeX,SizeY;
50 2Q BYTE Color;
51 lE {
52 tD3 /**/ verarbeitet eine Maske zu einer Farbe und /**/
53 xo /**/ kopiert sie auf das Ziel, ohne den Hinter- /**/

```

```

54 ZI /**/ grund zu verändern /**/
55 jN WORD i;
56 Mz UBYTE Minterm;
57 pA InitTmpBitMap (MaskBitMap);
58 WP for (i=0; i<DestBitMap->Depth; i++)
59 tM6 {
60 D6 if (Color & (1<<i))
61 L29 Minterm = 0xc0; /* Dest = Source OR Dest */
62 pc6 else
63 nw9 Minterm = 0x20; /* Dest = NOT Source AND Dest */
64 F86 BltBitMap (&TmpBitMap,MaskX,MaskY,DestBitMap,DestX,DestY,SizeX,SizeY,Minterm,1<<i,NULL);
65 3Y }
66 420 }
67 jT ReplaceColor (BitMap,X,Y,MaskBitMap,MaskX,MaskY,SizeX,SizeY,FromColor,ToColor)
68 GY struct BitMap *BitMap,*MaskBitMap;
69 Kz WORD X,Y,MaskX,MaskY,SizeX,SizeY;
70 3Y BYTE FromColor,ToColor;
71 5Y {
72 T73 /**/ Ersetzt eine Farbe durch eine andere /**/
73 AU GetColorMask (BitMap,X,Y,MaskBitMap,MaskX,MaskY,SizeX,SizeY,FromColor,NULL);
74 Zd CreateColor (MaskBitMap,MaskX,MaskY,BitMap,X,Y,SizeX,SizeY,ToColor,NULL);
75 D10 }
76 ZF ExchangeColor (BitMap,X,Y,MaskOBitMap,MaskOX,MaskOY,Mask1BitMap,Mask1X,Mask1Y,SizeX,SizeY,Color0,Color1)
77 B4 struct BitMap *BitMap,*MaskOBitMap,*Mask1BitMap;
78 Br WORD X,Y,MaskOX,MaskOY,Mask1X,Mask1Y,SizeX,SizeY;
79 V9 BYTE Color0,Color1;
80 Eh {
81 dd3 /**/ Vertauscht zwei Farben /**/
82 PO GetColorMask (BitMap,X,Y,MaskOBitMap,MaskOX,MaskOY,SizeX,SizeY,Color0,NULL);
83 jh GetColorMask (BitMap,X,Y,Mask1BitMap,Mask1X,Mask1Y,SizeX,SizeY,Color1,NULL);
84 5o CreateColor (MaskOBitMap,MaskOX,MaskOY,BitMap,X,Y,SizeX,SizeY,Color1,NULL);
85 Fv CreateColor (Mask1BitMap,Mask1X,Mask1Y,BitMap,X,Y,SizeX,SizeY,Color0,NULL);
86 Ot0 }
(C) 1987 M&T

```

Listing 1. Die Routine »BlitColor.h« binden Sie einfach mit »#include« in eigene Programme ein. Bitte mit dem Checksummer (Seite 159) eingeben.

Programmname:	Demo.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	s. Listing 1
Aufrufe:	s. Listing 1
Bemerkung:	Demo zeigt einige Möglichkeiten der Routine »BlitColor«

Programm : Demo.c

```

1 ks0 #include "exec/types.h"
2 Y1 #include "intuition/intuition.h"
3 NZ #include "graphics/text.h"
4 Up #include "graphics/gfx.h"
5 X5 #include "BlitColor.h"
6 Zs struct Screen *OpenScreen();
7 45 LONG GfxBase,
8 Ey5 IntuitionBase;
9 Rh0 struct RastPort *rp;
10 13 struct Screen *screen[3];
11 og struct NewScreen ns =
12 8b3 {
13 3P 0,0,320,256,0,
14 KP 0,1,0,
15 B6 CUSTOMSCREEN
16 ZV };
17 Wn0 static UWORD color [32] =
18 Eh3 {
19 gm 0x000, 0xffff, 0xff0, 0xfc0, 0xf90, 0xf60, 0xf30, 0xf00,

```



```

20 WW      0xf08, 0xf0c, 0xf0f, 0xc0f, 0xa0f, 0x70f, 0x40f, 0x00f,
21 hj      0x04f, 0x09f, 0x0bf, 0x0df, 0x0ff, 0x0fa, 0x0f0, 0x0e0,
22 yd      0x090, 0xba5, 0xb86, 0x866, 0xecc, 0x999, 0x666, 0x333
23 gc      };
24 llo #define RND RangeRand
25 jj      main ()
26 mp      {
27 hv3      WORD i;
28 oh      InitScreen ();
29 rm      /*** Bild zeichnen ***/
30 oy      for (i=0; i<1000; ++i)
31 ru6      {
32 cz      SetAPen (rp,RND(32));
33 hw      Move (rp,RND(320),12+RND(244));
34 fl      Draw (rp,RND(320),12+RND(244));
35 z4      }
36 va3      /*** Demo ***/
37 dr      Delay (50);
38 fh      for (i=0; i<100; i+=2)
39 lm6      ExchangeColor (&screen[0]->BitMap,0,12,&screen[1]->Bi
          itMap,0,12,&screen[2]->BitMap,0,12,160,244,1+RangeRan
          d(31),1+RangeRand(31));
40 gu3      Delay (50);
41 hf      for (i=0; i<31; ++i)
42 or6      ReplaceColor (&screen[0]->BitMap,0,12,&screen[1]->Bi
          tMap,0,12,320,244,1,1+1);
43 ld3      Free ();
44 ido }
45 uw      InitScreen ()
46 g9      {
47 bf3      WORD i;
48 px      if (! (GfxBase = OpenLibrary ("graphics.library",0)))
49 r86      Cleanup ("Can't open graphics.library");
50 sk3      if (! (IntuitionBase = OpenLibrary ("intuition.library",0
          )))
51 or6      Cleanup ("Can't open intuition.library");
52 xc3      for (i=2; i>=0; i--)

```

```

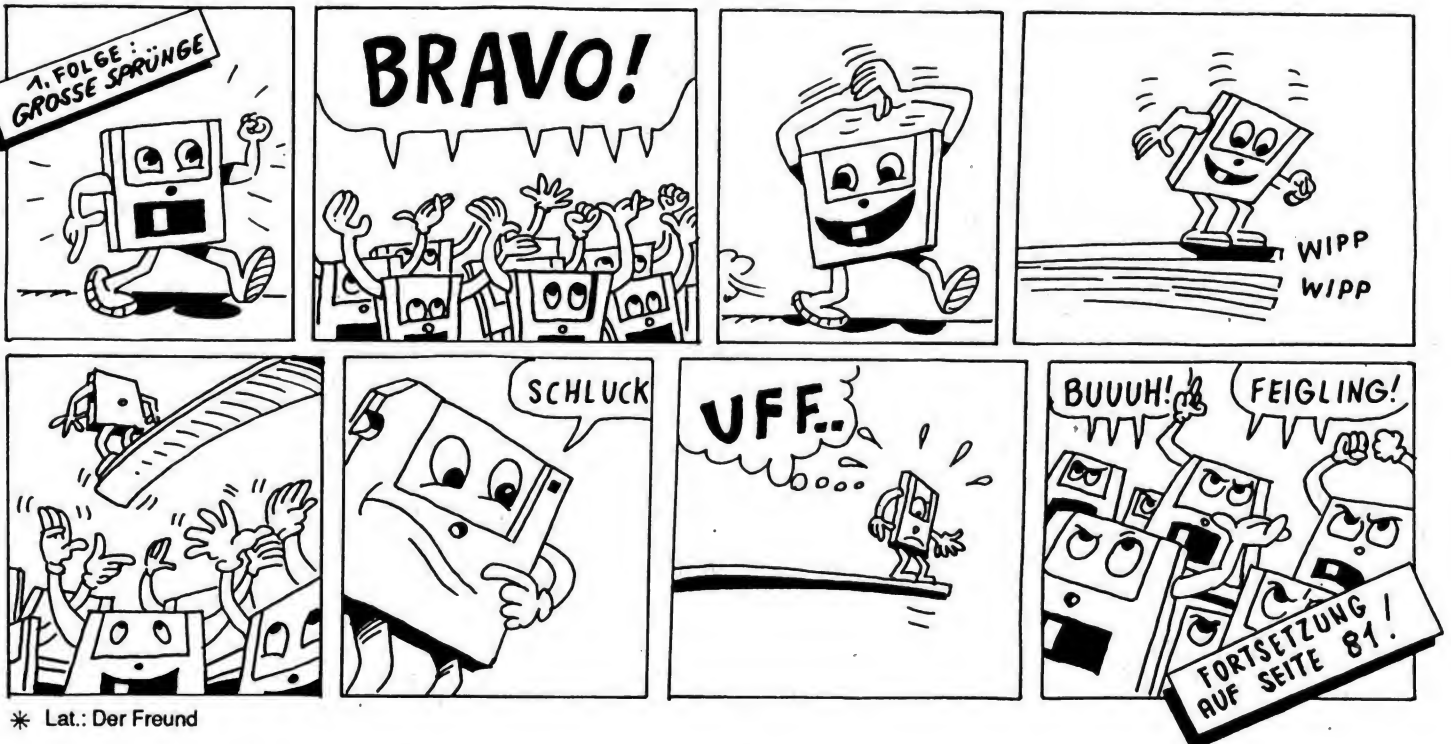
53 nG6      {
54 qu      if (i==0)
55 lQ9      ns.Depth = 5;
56 jW6      else
57 y99      ns.Depth = 1; /* Hintere Screens nur eine BitPlane
          */
58 Sj6      if (! (screen[i] = OpenScreen (&ns)))
59 cz9      Cleanup ("Can't open screen");
60 yT6      }
61 zm3      LoadRGB4 (&screen[0]->ViewPort,color,32);
62 nq      rp = &screen[0]->RastPort;
63 lW0 }
64 bG      Cleanup (str)
65 UX      char *str;
66 OT      {
67 UK3      printf ("%s\n",str);
68 kc      Free ();
69 Uk      Exit ();
70 8d0 }
71 jU      Free ()
72 6Z      {
73 lf3      WORD i;
74 kL      for (i=0; i<3; i++)
75 QM6      if (screen[i])
76 de9      CloseScreen (screen[i]);
77 W13      if (IntuitionBase)
78 JQ6      CloseLibrary (IntuitionBase);
79 533      if (GfxBase)
80 RK6      CloseLibrary (GfxBase);
81 Jo0 }
(C) 1987 M&T

```

Listing 2. »demo.c« zeigt Ihnen auf einfache Weise die Verwendung der Funktionen

AMICUS

DIE ABENTEUER EINER FLOPPY-DISC



* Lat.: Der Freund

Gravitation, die Massenanziehungskraft, ist für uns moderne Menschen so selbstverständlich, daß wir uns über sie kaum noch Gedanken machen. Doch gerade deshalb fehlt bei vielen das Verständnis für dieses faszinierende physikalische Phänomen.

Mit dem Programm »DeRev« (Listing 1), welches das Experimentieren mit der Gravitation in der Welt der Planeten und Sterne erlaubt, kann man sich nicht nur das fehlende Verständnis aneignen. Der Anwender ist zudem in der Lage, ein wenig den »Schöpfer« zu spielen und neue Welten und Sonnensysteme zu erschaffen und untergehen zu lassen (Bild 1).

Das Prinzip des Programms beruht darauf, bis zu hundert Objekte auf dem Bildschirm zu verteilen und jedem seine eigene Masse und Geschwindigkeit geben zu können. Wenn man die Berechnung startet, wird die Anziehungskraft jedes einzelnen Objektes auf jedes andere berechnet. Dadurch ändern sich die Bahnen und Geschwindigkeiten, womit sehr deutlich die Wirkung starker und schwacher Gravitationskräfte zu beobachten ist.

Um ein lauffähiges Programm zu erhalten, müssen die beiden Programmteile »DeRev.MAIN.c« und »DeRev.IO.c« mit dem Checksummer abgetippt und anschließend compiliert werden. Die Compileraufrufe für den Aztec C-Compiler lauten:

```
cc DeRev.MAIN.c +l -S
cc DeRev.IO.c
```

Anschließend sind die beiden *.o«-Dateien zusammenzulinken:

```
ln -o DeRev De-
Rev.MAIN.o DeRev.IO.o -
lm32 -lc32
```

Im Anschluß daran steht das lauffähige Programm unter dem Namen »DeRev« zur Verfügung. Das Programm benötigt die »mathtrans.library«, die im aktuellen »libs«-Verzeichnis stehen muß.

Wird das Programm gestartet, erscheint zunächst nur ein dunkelblauer Bildschirm. Mit der rechten Maustaste gelangt man zum Pull-Down-

Menü, mit dem das gesamte Programm gesteuert wird.

Es gibt insgesamt 19 Menüpunkte:

Start: Die Berechnung wird gestartet.

Stop: Die Berechnung wird angehalten.

Neues Objekt: Zu den schon bestehenden Objekten kann ein weiteres hinzugefügt werden. Dazu wird eine Reihe von einzelnen Requestern

von Proportional-Gadgets wurde verzichtet, da es kompliziert geworden wäre, mit diesen Gadgets sehr große Zahlenbereiche mit bis zu drei Stellen hinter dem Komma genau anzugeben. Ist die Position eingestellt, drückt man auf »OK«. Es erscheint ein Fenster, mit dem man die Masse des Objekts eingeben kann. Auch hier beendet »OK« die Funktion. Als letztes ist die Ge-

vorgenommen werden kann.

Position: Hier kann nachträglich die Position eines Objekts geändert werden.

Geschwindigkeit: Dient zur Änderung der Geschwindigkeit.

Löschen: Der Benutzer legt zuerst fest, welches Objekt gelöscht werden soll. Danach erscheint eine Sicherheitsabfrage. Wird diese mit »J« beantwortet, verschwindet das Objekt vom Bildschirm und wird auch nicht mehr in die Berechnung mit einbezogen. Wenn man nun zum Beispiel Objekt 2 löscht, entsteht an Stelle 2 aber kein Loch, sondern die nachfolgenden Objekte rücken in den Nummern um eins vor. Objekt 3 wird zu 2 und so weiter. Das gelöschte Objekt stellt sich nun aber an den frei gewordenen letzten Platz. Also kann man dieses Objekt wieder auf den Bildschirm bringen, indem man »Neues Objekt« anwählt und die Werte einfach nicht verändert. Nebenbei können auf diese Art und Weise auch Objekte nur für eine Weile aus der Berechnung ausgeschlossen werden.

Zeigen: Wenn man die Objekte eine Weile hat kreisen lassen, fällt es einem vielleicht schwer, unter den vielen Objekten und Bahnen noch festzustellen, welche Nummer welches Objekt hat. Deshalb kann in diesem Menüpunkt die Nummer des gesuchten Objekts angewählt werden. Drückt der Benutzer auf »OK«, wird das Objekt so lange blinken, bis die linke Maustaste gedrückt wird.

Zeitintervall: Bestimmt, wie viele Stunden zwischen zwei Berechnungsschritten vergehen sollen. Voreingestellt sind hier 24 Stunden, die ein guter Kompromiß zwischen Geschwindigkeit und Genauigkeit darstellen. Besonders bei stark exzentrischen Bahnen, also Bahnen, die stark von der

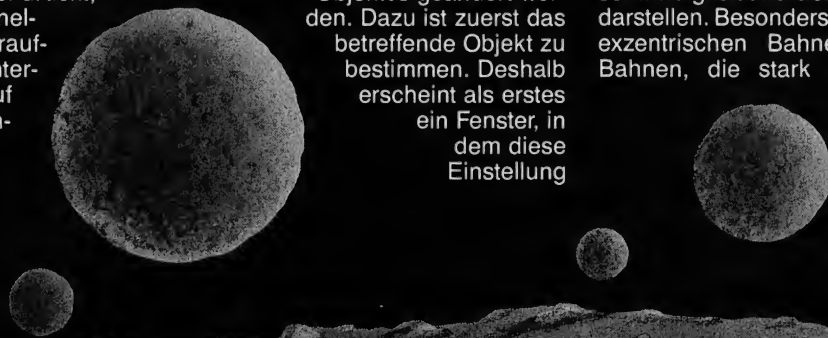
durchlaufen, in denen die Parameter einzustellen sind. Als erstes ist anzugeben, ob man das Objekt auf dem Bildschirm positionieren will. Wird »J« gewählt, kann mit der Maus über den Bildschirm gewandert und die gewünschte Position gewählt werden. Am oberen Bildschirmrand wird der Abstand von der Bildmitte in Millionen Kilometern angegeben. Nach Druck auf die linke Maustaste erscheint ein Fenster, in dem sich die Position noch einmal mit Hilfe von vier kreuzförmig angeordneten »Knöpfen« genauer angeben läßt. In dieses Fenster wäre man auch gekommen, hätte man sich am Anfang gegen die Positionierung auf dem Bildschirm entschieden. Je länger man auf einen Knopf drückt, um so schneller wird herauf- oder heruntergezählt. Auf die Verwen-

dschwindigkeit des Körpers einzustellen. Da sich das Objekt in beliebigen Richtungen über den Bildschirm bewegen kann, genügt es nicht, nur die Geschwindigkeit zu bestimmen. Vielmehr ist anzugeben, mit welcher Geschwindigkeit es sich in die X-Richtung (horizontal) und in die Y-Richtung (vertikal) bewegen soll. Natürlich können auch hier wie bei der Position negative Werte eingestellt werden. Ist auch diese letzte Eingabe mit »OK« abgeschlossen, erscheint das Objekt, wenn es nicht außerhalb des Bildschirms liegt, als kleiner weißer Punkt an der eingestellten Position.

Masse: Hiermit kann nachträglich, auch mitten in der Berechnung, die Masse eines Objektes geändert werden. Dazu ist zuerst das betreffende Objekt zu bestimmen. Deshalb erscheint als erstes ein Fenster, in dem diese Einstellung

Der Weltenbauer

Haben Sie nicht auch schon davon geträumt, Planeten zu erschaffen und auf eine Umlaufbahn um eine Sonne zu bringen? Erleben Sie mit unserer Simulation »hautnah« die Massenanziehungskraft bei Himmelskörpern.



Kreisform abweichen, machen sich bei großem Zeitintervall Rechenungenauigkeiten bemerkbar. Dies liegt nicht etwa an einer schlechten Programmierung des für die Bewegung zuständigen Programmteils oder an ungenauen Mathematikroutinen des C-Compilers, sondern an dem angewandten Rechenverfahren. Dem Programmator ist allerdings auch kein anderes Verfahren bekannt, welches hier Verwendung finden könnte. Wählt man den Zeitintervall genügend klein, werden auch sehr stark exzentrische Bahnen noch ziemlich genau berechnet.

Fixstern (JA/NEIN): Dieser Menüpunkt ist schon ein Schalter für sich. Bei jedem Anwählen verändert er sich. Aus »JA« wird »NEIN« und umgekehrt. In den meisten Fällen wird man die Bahnen von Planeten um einen Stern berechnen lassen. Da aber im allgemeinen die Massen der Planeten so gering sind, daß sie den Stern nicht sonderlich aus der Ruhelage bringen, kann man Zeit bei der Berechnung sparen, indem man »Fixstern (JA)« einstellt. Dann ist Objekt 1 unbeweglich geworden, fixiert, und es muß auf ein Objekt weniger die wirkenden Kräfte errechnet werden.

Das Programm geht davon aus, daß das Objekt 1 der Fixstern ist. Wählt man »Fixstern (NEIN)«, so wird nun auch der Stern beweglich. So kann man beispielsweise beobachten, was mit diesem und seinen Planeten geschieht, wenn ein sehr schweres Objekt vorbeifliegt.

Vergrößerung: In diesem Fenster kann festgelegt werden, wie viele Millionen Kilometer durch einen Bildpunkt repräsentiert werden. Je größer die Zahl, um so weiter entfernt man sich von den Objekten. Ihr Abstand zur Bildmitte wird kleiner, und man kann auch die äußeren Regionen eines Sonnensystems in Augenschein nehmen. Je kleiner die Zahl, um so größer werden die Bahnen dargestellt.

Uhr: Hier kann man bestimmen, ob am oberen Bildrand eine Uhr eingeblendet werden soll. Diese Uhr zeigt die verstrichenen Jahre und Tage an. Sollen neben den Jahren auch noch die Tage ausgegeben werden, hat dies den Nachteil, daß die ständige Ausgabe der Tage Zeit raubt und die Ablaufgeschwindigkeit so sinkt.

Gravo Grafik: Dieser Menüpunkt bietet die faszinierende Möglichkeit, die Stärke der

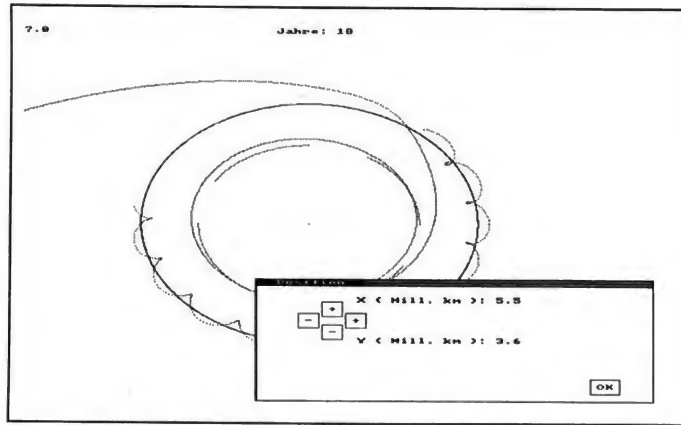


Bild 1. Ein Teil eines gedachten Sonnensystems mit den Umlaufbahnen der einzelnen Planeten

Gravitation bildlich darzustellen. Der Bildschirm wird dafür von oben nach unten abgetastet. In der dabei entstehenden Grafik wird die Stärke der Gravitationsfelder der Objekte durch Höhe und Breite von »Bergen«, den sogenannten Gravitationstrichtern, dargestellt. Zuerst muß ein Dehnungsfaktor bestimmt werden, der die Höhe der Berge beeinflusst. Dann ist anzugeben, welche Rechengenauigkeit gewünscht ist. Hier sind Werte zwischen 1 und 16 erlaubt. Bei 1 wird nur jede 16. Zeile des Bildschirms in die Rechnung mit einbezogen, bei 8 jede zweite und bei 16 alle. Die Zeit, die zum Erstellen einer Grafik benötigt wird, hängt maßgeblich von der gewählten Genauigkeit, aber auch von der Anzahl der Objekte ab. Hat man »Fixstern (JA)« aktiviert, wird Objekt 1 nicht in die Berechnung mit einbezogen. Dies hat den Vorteil, daß die kleinen Gravitationstrichter der Planeten, angesichts des gigantischen Trichters der Sonne, nicht völlig verschwinden. Die Berechnung der Grafik kann durch Betätigen des Close-Gadgets beendet werden.

Neuer Nullpunkt: Es kann der neue Bildschirmmittelpunkt mit dem Mauszeiger bestimmt werden. Damit wird ermöglicht, Objekte, die aus dem Bildschirm hinauswandern, weiter zu verfolgen.

Bildschirm löschen: Nach einer Sicherheitsabfrage wird der Bildschirm gelöscht.

Laden: Eine zuvor gespeicherte Konstellation wird nach einer Sicherheitsabfrage geladen und kann weiter berechnet werden. Es erscheint eine Zeile, in der der Pfad- oder Dateiname anzugeben ist.

Speichern: Alle Objekte im Speicher werden auf Diskette gesichert, auch diejenigen, die zur Zeit nicht berechnet wer-

den. Auch hier ist ein Dateiname anzugeben.

Neu: Nach einer Sicherheitsabfrage wird der Bildschirm gelöscht und die Anfangsparameter eingestellt.

Ende: Verlassen des Programms.

Tips für den Anfang

Damit man ein Gefühl für die richtigen Werte bekommt, ist unser Sonnensystem bereits voreingestellt. Das bedeutet,

wenn man ein paar mal »Neues Objekt« wählt und die Werte nicht verändert, erscheinen die Körper unseres Sonnensystems auf dem Bildschirm. Wenn man das fünfmal macht, hat man vor sich die Sonne und die inneren Planeten Merkur, Venus, Erde und Mars.

Hiermit kann man besonders gut experimentieren, wenn man beispielsweise ein sehr schweres Objekt erschafft und dieses durch das innere Sonnensystem »fliegen« läßt. Beobachten Sie dabei, was mit den Planeten geschieht.

Dazu läßt man jeden Planeten mindestens einen Umlauf vollenden, damit man sieht, wie die Bahnen für gewöhnlich aussehen. Dann wählt man »Neues Objekt«. Als nächstes käme eigentlich der Jupiter. Ihn positionieren wir aber ungefähr bei X=280 und Y=-100. Damit die Wirkung deutlicher wird, erhöhen wir die Masse um das 5fache bis 30fache. Starten Sie die Berechnung und sehen Sie, was passiert...

Aber experimentieren Sie selbst. So bekommen Sie am schnellsten eine Vorstellung über die Gravitation und es macht am meisten Spaß.

(J. Brendel/D. Meier/rs)

Programmname: DeRev

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Compiler: Aztec-C V3.4

Aufrufe: cc DeRev.MAIN.c +I -scc DeRev.IO.c
ln -o DeRevDeRev.MAIN.o
DeRev.IO.o -lm32 -lc32

Bemerkung: benötigt mathtrans.library

Programm : DeRev

```
1 DIO /*****
2 nJ ** De revolutionibus orbium coelestium **
3 sU ** ( DeRev.MAIN.c ) **
4 4Y ** (c) 1988 by **
5 fs ** Jürgen Brendel **
6 cI *****/
7 ih #include <math.h>
8 Wl #include <intuition/intuition.h>
9 Et #define MA 100 /* -- maximale Anzahl von Objekten -- */
10 JV #define M1 101
11 Su #define MENP 19 /* -- Anzahl der Menüpunkte -- */
12 JN int count,code,code2,class,Jahre=0,xx,yy,
13 Zc7 xalt[M1],yalt[M1],p[161];
14 q10 float gm[M1],m[M1],x[M1],y[M1],x2[M1],y2[M1],
15 yE7 Vx[M1],Vy[M1],Faktor,winkel,a,Vabl,xd,yd,
16 b5 Zeit,zeit2,gam,var2,fak,ystep,Tage=0.0;
17 r30 char string[40],buffer[71],undo[71],z,t,i,anz,start,
18 lJ7 bflag,vflag,uflag,tflag,jflag;
19 dJ0 extern save(),load();
20 Ho extern long *OpenLibrary();
21 9q extern struct Screen *OpenScreen();
```

Listing 1. Mit »DeRev.MAIN.c« erstellen Sie eigene Sonnensysteme, die Wirkung der Massenanziehung lernen Sie dabei »spielend« kennen. Bitte mit dem Checksummer (Seite 159) eingeben.

```

22 YI extern struct Window *OpenWindow();
23 qb struct Screen *screen;
24 7U struct Window *window, *window2;
25 h0 struct IntuitionBase *IntuitionBase;
26 pz struct GfxBase *GfxBase;
27 sP struct RastPort *rport, *rport2;
28 2R struct IntuiMessage *message;
29 LZ struct IntuiText mt[MENP];
30 As struct MenuItem mi[MENP];
31 2n struct Menu menu =
32 Sv {
33 VV2 NULL,10,0,186,10,MENUNENABLED,"(c) 1988 by J. Brendel",mi
34 rn0 };
35 op struct StringInfo strinf =
36 Wz {
37 gF2 (UBYTE *)&buffer,(UBYTE *)&undo,0,70,0,0,0,0,0,0,0,0
38 vr0 };
39 nw struct Gadget strgad =
40 a3 {
41 TP2 NULL,3,11,517,10,GADGHCOMP,GADGIMMEDIATE } RELVERIFY,
42 oT STRGADGET,NULL,0,NULL,0,&strinf,NULL,0
43 Ow0 };
44 XS struct NewScreen hauptscreen =
45 f8 {
46 AA2 0,0,640,512,2,0,1,HIRES } LACE,
47 LO CUSTOMSCREEN,NULL,NULL,NULL,NULL
48 510 };
49 LY struct NewWindow hauptwindow =
50 kd {
51 6j2 0,0,640,512,0,1, MENUPIK } MOUSEBUTTONS,
52 ts BORDERLESS } ACTIVATE } SMART_REFRESH,
53 MF NULL,NULL,NULL,NULL,NULL,10,10,640,512,CUSTOMSCREEN
54 B70 };
55 Wb struct NewWindow parawindow =
56 qJ {
57 er2 30,30,370,160,1,3,MOUSEBUTTONS,
58 u7 WINDOWDRAG } ACTIVATE,NULL,NULL,NULL,
59 v8 NULL,NULL,30,30,370,160,CUSTOMSCREEN
60 HDO };
61 AX struct NewWindow gragrawindow =
62 wP {
63 BZ2 30,0,520,512,1,3,CLOSEWINDOW,
64 i5 WINDOWCLOSE } WINDOWDRAG } ACTIVATE } GIMMEZEROZERO,
65 Md NULL,NULL,(UBYTE *)" GravoGrafik ",
66 jX NULL,NULL,30,0,520,512,CUSTOMSCREEN
67 OK0 };
68 88 struct NewWindow lswindow =
69 3W {
70 rG2 0,0,521,22,1,3,GADGETUP,WINDOWDRAG } ACTIVATE,
71 3Q &strgad,NULL,NULL,NULL,NULL,0,0,520,13,CUSTOMSCREEN
72 TP0 };
73 NN2 /* ----- */
74 Od1 /* ---- Funktionen ---- */
75 PPO /* ----- */
76 qF void openall()
77 Be {
78 Ap2 /* ---- Libraries, Screen und Window öffnen ---- */
79 xr if(!(IntuitionBase = (struct IntuitionBase *)
80 F24 OpenLibrary("intuition.library", 0))) exit(FALSE);
81 vJ2 if(!(GfxBase = (struct GfxBase *)
82 3d4 OpenLibrary("graphics.library", 0)))
83 qZ2 { CloseLibrary(IntuitionBase); exit(FALSE); }
84 tc if(!(screen = OpenScreen(&hauptscreen)))
85 he { CloseLibrary(IntuitionBase); CloseLibrary(GfxBase);
86 aV4 exit(FALSE); }
87 wF2 hauptwindow.Screen = parawindow.Screen =
88 oR7 gragrawindow.Screen = lswindow.Screen = screen;
89 KO2 if(!(window = OpenWindow(&hauptwindow)))
90 B1 { CloseScreen(screen); CloseLibrary(IntuitionBase);
91 HP4 CloseLibrary(GfxBase); exit(FALSE); }
92 TJ2 rport=window->RPort;
93 jV /* ---- Menüs installieren ---- */
94 bb for(t=0; t<MENP; t++)
95 Tw {
96 7v4 mi[t].NextItem = (t<(MENP-1)) ? &mi[t+1] : NULL;
97 ss mi[t].LeftEdge = 0; mi[t].TopEdge = t*16;
98 QA mi[t].Width = 178; mi[t].Height = 10;
99 67 mi[t].Flags = ITEMTEXT | ITEMENABLED | HIGHCOMP;
100 em mi[t].MutualExclude = NULL;
101 iy mi[t].ItemFill = (APTR) &mt[t];
102 c1 mi[t].Command = NULL; mi[t].SubItem = NULL;
103 Ks mi[t].NextSelect = NULL;
104 JS mt[t].FrontPen = 0; mt[t].BackPen = 1;
105 O2 mt[t].TopEdge = 1; mt[t].LeftEdge = 5;

```

```

106 uP mt[t].DrawMode = JAM2; mt[t].ITextFont = NULL;
107 kt mt[t].NextText = NULL;
108 kF2 }
109 yI mt[0].IText = "Start";
110 NA mt[1].IText = "Stop";
111 1M mt[2].IText = "Neues Objekt";
112 wB mt[3].IText = "Masse";
113 40 mt[4].IText = "Position";
114 Rg mt[5].IText = "Geschwindigkeit";
115 9g mt[6].IText = "Löschen";
116 Jn mt[7].IText = "Zeigen";
117 uv mt[8].IText = "Zeitintervall";
118 OD mt[9].IText = "Fixstern (JA)";
119 S7 mt[10].IText = "Vergrößerung";
120 fZ mt[11].IText = "Uhr";
121 cX mt[12].IText = "Gravo - Grafik";
122 wV mt[13].IText = "Neuer Nullpunkt";
123 Nt mt[14].IText = "Bildschirm löschen";
124 4C mt[15].IText = "Laden";
125 En mt[16].IText = "Speichern";
126 3S mt[17].IText = "Neu";
127 vh mt[18].IText = "Ende";
128 bb SetMenuStrip(window, &menu);
129 6n SetRGB4(&(screen->ViewPort), 0, 0, 0, 7);
130 PN SetRGB4(&(screen->ViewPort), 1, 3, 15, 3);
131 Qw SetRGB4(&(screen->ViewPort), 2, 15, 15, 15);
132 bL SetRGB4(&(screen->ViewPort), 3, 15, 0, 0);
133 9e0 }
134 Do void startwerte()
135 7a {
136 zA2 for(t=0; t<MA; t++)
137 qx { m[t]=1.0e6; x[t]=0.0; y[t]=0.0; Vx[t]=0.0; Vy[t]=0.0; }
138 Zx m[1]=1.989e15; x[1] = 0.0; y[1] = 0.0; /* Sonne */
139 BMG Vx[1] = 0.0; Vy[1] = 0.0;
140 1t2 m[2]=0.333e09; x[2] = 57.9; y[2] = 0.0; /* Merkur */
141 6KG Vx[2] = 0.0; Vy[2] = 47.9;
142 rG2 m[3]=4.870e09; x[3] = 108.2; y[3] = 0.0; /* Venus */
143 aOG Vx[3] = 0.0; Vy[3] = 35.0;
144 Wo2 m[4]=5.977e09; x[4] = 149.6; y[4] = 0.0; /* Erde */
145 PiG Vx[4] = 0.0; Vy[4] = 29.765;
146 rM2 m[5]=0.644e09; x[5] = 227.9; y[5] = 0.0; /* Mars */
147 qCG Vx[5] = 0.0; Vy[5] = 24.1;
148 Im2 m[6]=1.899e12; x[6] = 778.3; y[6] = 0.0; /* Jupiter */
149 rIG Vx[6] = 0.0; Vy[6] = 13.1;
150 TY2 m[7]=5.684e11; x[7] = 1427.0; y[7] = 0.0; /* Saturn */
151 DzG Vx[7] = 0.0; Vy[7] = 9.6;
152 Po2 m[8]=8.676e10; x[8] = 2870.0; y[8] = 0.0; /* Uranus */
153 GAG Vx[8] = 0.0; Vy[8] = 6.9;
154 OS2 m[9]=1.029e11; x[9] = 4496.0; y[9] = 0.0; /* Neptun */
155 DxG Vx[9] = 0.0; Vy[9] = 5.4;
156 M12 start=2; mt[9].IText = "Fixstern (JA)"; uflag=0;
157 RZ Faktor=1.7e9; anz=0; Zeit=86400.0; zeit2=Zeit/3600.0;
158 NL fak=5000000.0; ystep=1.0; gam=.00667*(Zeit/1.0e5);
159 Z40 }
160 k4 void wertaufbereiten()
161 XO {
162 9x2 gm[t]=gam*m[t];
163 cw x2[t]=(x[t]*=1.0e9); y2[t]=(y[t]*=1.0e9);
164 5E Vx[t]*=(Zeit*1000.0); Vy[t]*=(Zeit*1000.0);
165 EO xalt[t]=320+(int)(x[t]/Faktor);
166 lq yalt[t]=256+(int)(y[t]/Faktor);
167 hC0 }
168 Vo void ifuflag()
169 f8 {
170 N62 if(uflag == 1)
171 hA {
172 1Q4 SetAPen(rport, 2);
173 uc if(tflag==1)
174 kD {
175 2Q6 sprintf(string,"Tage: %.1f ",Tage);
176 h5 Move(rport, 0, 8);
177 Qc Text(rport, (int)string, (int)(strlen(string)));
178 sN4 }
179 WJ sprintf(string,"Jahre: %ld ",Jahre);
180 oQ Move(rport, 290, 8);
181 Ug Text(rport, (int)string, (int)(strlen(string)));
182 wR2 }
183 xS0 }
184 KV void clearsreen(neum)
185 bS char neum;
186 wP {
187 c62 Move(rport, 0, 0); ClearScreen(rport); ifuflag();
188 ro SetAPen(rport,2);
189 IA if(neum == 1)

```



```

190 st4      for(t=1; t<=anz; t++)
191 P06        WritePixel(rport, xalt[t], yalt[t]);
192 6b0 }
193 JH void print(s,aa,bb)
194 N9 char *s; int aa,bb;
195 5Y {
196 3I2      Move(rport2, (int)aa, (int)bb);
197 uC      Text(rport2, (int)s, (int)(strlen(s)));
198 Ch0 }
199 dI void rec(s,aa,bb)
200 TF char *s; int aa,bb;
201 Be {
202 tW2      int breite=19;
203 a0      if(aa == -1)
204 Dg      { aa=310; bb=130; breite=27; s="OK"; }
205 wJ      Move(rport2, (int) aa, (int) bb);
206 iP      Draw(rport2, (int)(aa+breite), (int) bb);
207 tL      Draw(rport2, (int)(aa+breite), (int)(bb+19));
208 65      Draw(rport2, (int) aa, (int)(bb+19));
209 xB      Draw(rport2, (int) aa, (int) bb);
210 E3      print(s,aa+6,bb+12);
211 Pu0 }
212 mI void button()
213 Nq {
214 ok2      message = (struct IntuiMessage *)GetMsg(window2->UserPor
t);
215 FN      code=message->Code; class=message->Class;
216 gs      for(count=0; count<8000; count++);
217 OR      if(class == MOUSEBUTTONS)
218 Sv      {
219 LI4      if(code == SELECTUP) bflag=0;
220 eT      if(code == SELECTDOWN) bflag=1;
221 Z42 }
222 a50 }
223 yn char booltest(wtitle, text)
224 gM char *wtitle, *text;
225 Z2 {
226 742      parawindow.Title = wtitle; parawindow.Height = 110;
227 f6      rport2=(window2=OpenWindow(&parawindow))->RPort;
228 o9      vflag = 2; SetAPen(rport2, 2);
229 3f      rec("J",40,30); rec("N",40,60); SetAPen(rport2, 3);
230 pU      sprintf(string, "%s", text); print(string, 88, 57);
231 P7      while(vflag == 2)
232 g9      {
233 904      button(); vflag = 2;
234 sI      if(bflag == 1)
235 jC      {
236 jW6      xx=window2->MouseX; yy=window2->MouseY;
237 Pw      if(xx>=40 && xx<=59 && yy>=30 && yy<=49) vflag=1;
238 kM      if(xx>=40 && xx<=59 && yy>=60 && yy<=79) vflag=0;
239 rM4      }
240 sN2 }
241 df      for(count=0; count<10000; count++);
242 wC      CloseWindow(window2); parawindow.Height = 160;
243 f9      bflag = 0; return vflag;
244 wR0 }
245 HZ float einst(kr, wtitle, text, schritt, sfak, sfak2, var, max, min, s
max, kom, exp)
246 sG char kr, *wtitle, *text, kom, exp;
247 Xo float schritt, sfak, sfak2, var, max, min, smax;
248 wP {
249 UP2      float schritt2=schritt; int fl=0; char exflag;
250 HL      if(kr > 0) fl=22; if(kr == 1) text=" ( km/s ) : ";
251 JT      if(kr == 2) text=" ( Mill. km ) : ";
252 iI      parawindow.Title = wtitle;
253 5W      rport2=(window2=OpenWindow(&parawindow))->RPort;
254 iJ      SetAPen(rport2, 2); rec("-", -1, 0);
255 3N      rec("+", 40+f1, 30); rec("-", 40+f1, 60);
256 HJ      if(kr > 0) { rec("-", 40, 45); rec("+", 84, 45); }
257 H1      SetAPen(rport2, 3); exflag=0;
258 Hs      while(exflag == 0)
259 7a      {
260 Zx4      button(); vflag=0;
261 Jj      if(bflag == 1)
262 Ad      {
263 Ax6      xx=window2->MouseX; yy=window2->MouseY;
264 iQ      if(xx>=40+f1 && xx<=59+f1 && yy>=30 && yy<=49)
265 zO8      { var+=(schritt*sfak); vflag=1; }
266 X26      if(xx>=40+f1 && xx<=59+f1 && yy>=60 && yy<=79)
267 588      { var+=(schritt*sfak); vflag=1; }
268 WJ6      if(xx>=310 && xx<=337 && yy>=130 && yy<=159)
269 jk8      exflag=1;
270 Dj6      if(kr > 0)
271 Jm      {

```

```

272 G08      if(xx>=40 && xx<=59 && yy>=46 && yy<=65)
273 CnA      { var2+=(schritt*sfak); vflag=1; }
274 Ar8      if(xx>=84 && xx<=103 && yy>=46 && yy<=65)
275 8hA      { var2+=(schritt*sfak); vflag=1; }
276 Sx6      }
277 Ty4      }
278 s3      if(var > max) { var = min; schritt = schritt2; }
279 rC      if(var < min) { var = max; schritt = schritt2; }
280 7T      if(var2 > max) { var2 = min; schritt = schritt2; }
281 wG      if(var2 < min) { var2 = max; schritt = schritt2; }
282 VS      if(exp == 1 && schritt == schritt2)
283 Vy      {
284 1S6      schritt=1.0;
285 lg      while( (schritt*=10.0) < var ); schritt/=1.0e4;
286 c74      }
287 7o      if(kom > 0 && exp == 0)
288 a3      {
289 KU6      if(kom == 2)
290 vu8      sprintf(string, "%s%.3f", text, var/sfak);
291 WJ6      else
292 e7      {
293 5U8      if(kr == 0)
294 BBA      sprintf(string, "%s%.1f", text, var/sfak);
295 DY8      if(kr == 1)
296 yXA      sprintf(string, "%s%.3f", text, var/sfak);
297 Lc8      if(kr == 2)
298 uR8      sprintf(string, "%s%.1f", text, var/sfak);
299 pK6      }
300 qL4      }
301 gT      else
302 HG6      sprintf(string, "%s%.0f", text, var/sfak);
303 OW4      if(exp == 1)
304 MN6      sprintf(string, "%s%.3e", text, var/sfak);
305 Mr4      if(kr == 0) print(string, 88, 57);
306 lY      else
307 tM      {
308 uu6      print(string, 95, 83);
309 Rm      if(kr == 1)
310 Vm8      sprintf(string, "%s%.3f", text, var2/sfak);
311 Zq6      if(kr == 2)
312 Rg8      sprintf(string, "%s%.1f", text, var2/sfak);
313 YQ6      print(string, 95, 30);
314 4Z4      }
315 nX      if(vflag == 1)
316 2V      {
317 SZ6      schritt*=sfak2;
318 gj      if(schritt> smax) schritt=smax;
319 2m      if(( kr>0 && (xx>=40+f1 && xx<=59+f1) ) ) kr == 0
320 6Z      {
321 tt8      if(schritt*sfak > abs(var)/5.0 && yy>=60)
322 wZA      schritt=abs(var)/5.0/sfak;
323 Di6      }
324 3q      else
325 yZ8      if(schritt*sfak > abs(var2)/5.0 && xx<=40+f1)
326 rAA      schritt=abs(var2)/5.0/sfak;
327 Hm4      }
328 7u      else
329 Fi      {
330 Bt6      schritt=schritt2;
331 Sy      if(exp == 1)
332 iI      {
333 oF8      schritt=1.0;
334 oT      while( (schritt*=10.0) < var ); schritt/=1.0e4;
335 Pu6      }
336 Qv4      }
337 Rw2      }
338 qQ      CloseWindow(window2); bflag=0; return var;
339 Ty0 }
340 EI void lsgadwart()
341 Ru {
342 GH2      window2 = OpenWindow(&lswindow);
343 Tv      class = GADGETDOWN;
344 59      while(class != GADGETUP)
345 Vy      {
346 ws4      message = (struct IntuiMessage *)GetMsg(window2->UserP
ort);
347 vL      class = message->Class;
348 c72      }
349 KY      strcpy(undo, buffer); CloseWindow(window2);
350 e90 }

```

Listing 1. (Fortsetzung)

```

351 u4 void posinp(ob)
352 JV int ob;
353 d6 {
354 kz2 char naeheflag=1; int bp;
355 so bp = (int)(booltest(" Achtung! ",
356 AXE "Bildschirmposition? (J/N)"));
357 Xa2 for(count=1; count<10000; count++);
358 BW while(naeheflag == 1)
359 JC {
360 Lf4 if(bp == 1)
361 LE {
362 dA6 code = 0; class = 0; SetAPen(rport, 2);
363 Xq while(! (class == MOUSEBUTTONS && code == SELECTUP))
364 oH {
365 328 message = (struct IntuiMessage *)GetMsg(window->Us
erPort);
366 WQ class=message->Class; code=message->Code;
367 uB xd=(float)((window->MouseX)-320);
368 Oa yd=(float)((window->MouseY)-256);
369 92 sprintf(string, " X ( Millionen km ) : %.1f ",
xd*Faktor/1.0e9);
370 aIF Move(rport, 0, 8);
371 qE8 Text(rport, (int)string, (int)(strlen(string)));
372 Z1 sprintf(string, " Y ( Millionen km ) : %.1f ",
yd*Faktor/-1.0e9);
373 mc Move(rport, 290, 8);
374 Text(rport, (int)string, (int)(strlen(string)));
375 xZ8 }
376 dp SetAPen(rport,0); RectFill(rport, 0, 0, 640, 9);
377 5a6 ifuflag();
378 ft }
379 jM else
380 8d4 { xd=x[ob]/Faktor; yd=y[ob]/Faktor; }
381 yI var2=xd*Faktor/1.0e9;
382 6S y[ob] = 1.0e9*einst(2, " Position ", "", 0.1, 1.0,
383 Xh 1.1, yd*Faktor/-1.0e9, 1.0e6, -1.0e6, 100000.0, 1.0);
384 bk x[ob]=var2*1.0e9; y[ob]=-y[ob]; naeheflag=0;
385 Us5 for(t=1; t<anz; t++)
386 Y64 if(abs(x[ob]-x[t]) < 100000000.0 && t!=ob &&
387 VI abs(y[ob]-y[t]) < 100000000.0) naeheflag = 1;
388 Eh6 }
389 ap8 SetAPen(rport, 01);
390 In2 WritePixel(rport, xalt[ob], yalt[ob]);
391 kL }
392 b3 void tausch(xt,yt)
393 Lq0 int xt,yt;
394 VS {
395 rZ x[xt]=x[yt]; y[xt]=y[yt];
396 Kn x2[xt]=x2[yt]; y2[xt]=y2[yt];
397 sS2 xalt[xt]=xalt[yt]; yalt[xt]=yalt[yt];
398 W0 Vx[xt]=Vx[yt]; Vy[xt]=Vy[yt];
399 S2 m[xt]=m[yt]; gm[xt]=gm[yt];
400 4s }
401 61 /* ----- */
402 Uz0 /* --- Menü - Punkte --- */
403 LM2 /* ----- */
404 3D1 void bewegung()
405 NOO {
406 Tq register char tt,ii;
407 Vy while(class != MENUPIK)
408 HQ2 {
409 oA message = (struct IntuiMessage *)GetMsg(window->UserPo
410 Y1 rt);
411 nm4 class = message->Class;
412 y0 if(uflag == 1)
413 I1 {
414 c5 SetAPen(rport, 2); Tage+=zeit2/24.0;
415 JO6 while(Tage>365.0) { Tage-=365.0; Jahre++; jflag=1; }
416 o9 if(Jahre>2000000000) Jahre=0;
417 1B if(zeit2 < 8760.0 && tflag==1)
418 IJ {
419 hA sprintf(string, "Tage: %.1f ", Tage);
420 zN8 Move(rport, 0, 8);
421 e2 Text(rport, (int)string, (int)(strlen(string)));
422 NZ }
423 pK6 if(jflag == 1)
424 a8 {
425 nG sprintf(string, "Jahre: %ld ", Jahre);
426 V18 Move(rport, 290, 8); jflag=0;
427 DU Text(rport, (int)string, (int)(strlen(string)));
428 Tf }
429 vQ6 }
430 wR4 }

```

```

431 Wa for(tt=start; tt<=anz; tt++)
432 ty { x2[tt]=x[tt]; y2[tt]=y[tt]; }
433 Yc for(tt=start; tt<=anz; tt++)
434 wP {
435 Z16 SetAPen(rport,3); WritePixel(rport,xalt[tt],yalt[tt])
;
436 bq xalt[tt]=320+(int)(x[tt]/Faktor);
437 2R yalt[tt]=256+(int)(y[tt]/Faktor);
438 WJ SetAPen(rport,2); WritePixel(rport,xalt[tt],yalt[tt])
;
439 mg y[tt]-=Vy[tt]; x[tt]-=Vx[tt];
440 XE for(ii=1; ii<=anz; ii++)
441 3W {
442 qM8 if(ii!=tt)
443 5Y {
444 KtA xd=x[tt]-x2[ii]; yd=y[tt]-y2[ii]+1.0e-15;
445 XD winkel=atan(yd/xd); xd/=1.0e6; yd/=1.0e6;
446 jX Vabl = (xd>0.0) ? (gm[ii]/(xd*xd+yd*yd)) :
447 IMS (-gm[ii]/(xd*xd+yd*yd));
448 NdA Vx[tt]+=cos(winkel)*Vabl*Zeit;
449 zp Vy[tt]+=sin(winkel)*Vabl*Zeit;
450 G18 }
451 Hm6 }
452 In4 }
453 Jo2 }
454 Kp0 }
455 zD void neu()
456 I1 {
457 7T2 float ber=1; posinp((int)(++anz));
458 K3 while( (ber*=10.0) < m[anz] ); ber/=1.0e10;
459 CJ m[anz] = einst(0, " Masse ", "Trillionen Tonnen: ",
460 Mb8 ber, 1.0, 1.15, m[anz]/1.0e6, 1.0e12, 0.00001, 1.0e11, 0, 1
);
461 nN2 m[anz]*=1.0e6; gm[anz]=gam*m[anz];
462 Q4 Vy[anz]/=Zeit; var2=Vx[anz]/Zeit/-1000.0;
463 na Vy[anz] = einst(1, " Geschwindigkeit ", "", 0.001,
464 3S6 1.0, 1.15, Vy[anz]/1000.0, 1.0e4, -1.0e4, 1000.0, 1.0);
465 9t2 Vx[anz]=var2*Zeit*-1000.0; Vy[anz]*=1000.0*Zeit;
466 xG x2[anz]=x[anz]; y2[anz]=y[anz];
467 EY xalt[anz]=320+(int)(x[anz]/Faktor);
468 oI yalt[anz]=256+(int)(y[anz]/Faktor);
469 ab SetAPen(rport, 2); WritePixel(rport,xalt[anz],yalt[anz]);
470 a50 }
471 RA void masse()
472 Y1 {
473 xY2 float ber=1;
474 sA int obn = (int)(einst(0, " Masse ", "Objekt: ",
475 975 1.0, 1.0, 1.0, 1.0, (float)anz, 1.0, 1.0, 0.0));
476 CJ2 while( (ber*=10.0) < m[obn] ); ber/=1.0e10;
477 Gb m[obn] = einst(0, " Masse ", "Trillionen Tonnen: ",
478 kp7 ber, 1.0, 1.15, m[obn]/1.0e6, 1.0e12, 0.00001, 1.0e11, 0, 1
);
479 pj2 m[obn]*=1.0e6; gm[obn]=gam*m[obn];
480 kF0 }
481 3u void position()
482 1B {
483 342 int obn = (int)(einst(0, " Position ", "Objekt: ",
484 IG5 1.0, 1.0, 1.0, 1.0, (float)anz, 1.0, 1.0, 0.0));
485 HA2 posinp(obn); SetAPen(rport,2); x2[obn]=x[obn]; y2[obn]=y[
obn];
486 8X WritePixel(rport, (xalt[obn]=320+(int)(x[obn]/Faktor)),
487 yp4 (yalt[obn]=256+(int)(y[obn]/Faktor)));
488 sN0 }
489 oS void geschw()
490 qJ {
491 DW2 int obn = (int)(einst(0, " Geschwindigkeit ", "Objekt: ",
492 Q05 1.0, 1.0, 1.0, 1.0, (float)anz, 1.0, 1.0, 0.0));
493 5L2 Vy[obn]/=Zeit; var2=Vx[obn]/Zeit/-1000.0;
494 sT Vy[obn] = einst(1, " Geschwindigkeit ", "", 0.001,
495 KX6 1.0, 1.15, Vy[obn]/1000.0, 1.0e4, -1.0e4, 1000.0, 1.0);
496 K12 Vx[obn]=var2*Zeit*-1000.0; Vy[obn]*=1000.0*Zeit;
497 1W0 }
498 11 void loesch()
499 zS {
500 wH2 int obn = (int)(einst(0, " Löschen ", "Objekt: ",
501 ZX5 1.0, 1.0, 1.0, 1.0, (float)anz, 1.0, 1.0, 0.0));
502 JN2 if(((int)(booltest(" Achtung! ",
503 UaL "Wirklich löschen? (J/N)"))) == 1)
504 4X2 {
505 Hu4 if(obn==1) { start=1; mt[9].IText="Fixstern (NEIN)"; }
506 eP tausch(M1,obn); SetAPen(rport,0);
507 7N WritePixel(rport,xalt[obn],yalt[obn]);
508 Sy if(obn < anz--)
509 9c {

```



```

510 046      for(t=obn; t<=anz; t++)
511 vA8          tausch(t,t+1);
512 Vw6          tausch(anz+1,M1);
513 Hm4      }
514 In2      }
515 Jo0      }
516 rJ      void zeig()
517 Hk      {
518 ts2          int obn = (int)(einst(0," Zeigen ", "Objekt: ",
519 2E5              1.0,1.0,1.0,1.0,(float)anz,1.0,1.0,0.0)); t=0;
520 HJ2          for(count=0; count<5500; count++)
521 Lo              {
522 sa4                  SetAPen(rport, (int)t); t++; if(t>3) t=0;
523 6u                  WritePixel(rport, xalt[obn], yalt[obn]);
524 Sx2              }
525 pt          SetAPen(rport,2); WritePixel(rport,xalt[obn],yalt[obn]);
526 Uz0      }
527 5k      void zeit()
528 Sv      {
529 q02          for(t=1; t<MA; t++) { Vx[t]/=Zeit; Vy[t]/=Zeit; }
530 Pz          Zeit = einst(0," Zeit ", "Zeit (h): ",0.1,3600.0,
531 R46              1.1,Zeit,9.0e7,0.09,1000.0,1.0);
532 Pn2          for(t=1; t<MA; t++) { Vx[t]*=Zeit; Vy[t]*=Zeit; }
533 Oj          gam=-0.0667*(Zeit/1.0e5); zeit2=Zeit/3600.0;
534 zz          for(t=1; t<MA; t++) gm[t]=gam*m[t];
535 d80      }
536 C5      void vergr()
537 b4      {
538 Fu2          Faktor = einst(0," Vergrößerung ", "Mill. km pro Punkt:
539 qD7              ",0.1,
540 z42              1.0,1.1,Faktor/1.0e9,1000.0,0.0,100.0,1.0);
541 Z1          Faktor*=1.0e9; SetAPen(rport, 3);
542 pL          for(t=1; t<MA; t++)
543 qQ4              { xalt[t]=320+(int)(x[t]/Faktor);
544 mH0                  yalt[t]=256+(int)(y[t]/Faktor); }
545 cx      void uhr()
546 kD      {
547 eH2          if(((int)(booltest(" Uhr ", "Uhr? (J/N)")) == 1)
548 mF              {
549 7R4              uflag=1; tflag=0;
550 s8              if(((int)(booltest(" Uhr ",
551 23B                  "Auf null setzen? (J/N)")) == 1)
552 hw4                  { Tage=0.0; Jahre=0; }
553 m5              if(((int)(booltest(" Uhr ", "Tage? (J/N)")) == 1)
554 Yu6                  tflag=1;
555 Pf4              SetAPen(rport,0); RectFill(rport,0,0,640,9);
556 nk              SetAPen(rport,2);
557 6o              if(tflag==1)
558 wP              {
559 Ec6                  sprintf(string,"Tage: %.1f ",Tage);
560 tH                  Move(rport, 0, 8);
561 co                  Text(rport, (int)string, (int)(strlen(string)));
562 4Z4              }
563 hu              sprintf(string,"Jahre: %ld ",Jahre);
564 Qh              Move(rport, 290, 8); jflag=0;
565 gs              Text(rport, (int)string, (int)(strlen(string)));
566 8d2              }
567 y1              else
568 6Z              {
569 R04                  SetAPen(rport, 0); RectFill(rport, 0, 0, 640, 9);
570 n9                  uflag=0;
571 Di2              }
572 EJO      }
573 sd      void gragraf()
574 Cf      {
575 d02          register char ii; float y0,x0,v,x1,y1;
576 ns          register int yyy,xt,yt=425,xstart=96;
577 nP          fak = einst(0," Vergrößerung ", "Dehnungsfaktor: ",1.0,
578 JW5              1.0,1.5,fak,500000000.0,0.1,10000000.0,0.0);
579 gV2          ystep = 16.0/einst(0," Genauigkeit ",
580 203              "Schritte pro Zeile: ",1.0,1.0,1.0,ystep,16.0,1.0,1.0,0,
581 5v2              0);
582 Kn          if(window2=OpenWindow(&gragrawindow))
583 w64          {
584 KH              rport2=window2->RPort;
585 Nq              for(yyy=1; yyy<=512; yyy+=16)
586 5y6                  {
587 N08                      if((struct IntuiMessage *)GetMsg(window2->UserPort))
588 Yd6                          goto gragraschluss;
589 hQ                      xl=(float)(2*xstart); yt+=2; yl=(float)yt;
590 Sv                      for(y0=(float)yyy; y0<=(float)yyy+16.0; y0+=ystep)
591 I48                          {

```

```

592 kJ          for(x0=1.0; x0<=640.0; x0+=4.0)
593 Vy          {
594 L5A              v=0.0;
595 N2              for(ii=start; ii<=anz; ii++)
596 Y1              {
597 sbC                  if(v < 5000.0)
598 a3                  {
599 FLE                      xd=(x0-(float)320)*Faktor-x[ii]+1.0e-18;
600 Ej                      yd=(y0-(float)256)*Faktor-y[ii];
601 3J                      winkel=atan(yd/xd); xd/=1.0e6; yd/=1.0e6;
602 OG                      v+=(gm[ii]/(xd*xd+yd*yd))*fak/1.0e4;
603 JEC                  }
604 kFA              }
605 2C              if(v > 800.0) v=800.0;
606 yr              p[xt]=(p[xt++]+(int)v)/2;
607 nI8              }
608 oJ6              }
609 Q1              for(xt=1, x0=1.0; x0<=640.0; x0+=4.0)
610 mF              {
611 WD8                  SetAPen(rport2, 0);
612 Sb                  Move(rport2, (int)(xl+1.0), yt+1);
613 kO                  Draw(rport2, (int)(xl+1.0), yt-p[xt]+1);
614 OR                  Move(rport2, (int)xl, (int)yl);
615 Xb                  xl=2.0*((float)xstart+x0/4.0); yl=(float)(yt-p[xt]);
616 DI                  SetAPen(rport2, 3); Draw(rport2,(int)xl,(int)yl);
617 zR                  p[xt++]=0;
618 yT6              }
619 L8              Draw(rport2, (int)xl, yt); xstart-=3;
620 OV4              }
621 gg              Wait(1 << window2->UserPort->mp_SigBit);
622 xW0          gragraschluss:
623 r84              CloseWindow(window2);
624 4Z2              }
625 23              ystep=16.0/ystep;
626 6b0      }
627 8J      void neunull()
628 4X      {
629 rf2          class = 0; SetAPen(rport, 2);
630 q9          while(!(class == MOUSEBUTTONS && code == SELECTUP))
631 7a          {
632 ML4              message = (struct IntuiMessage *)GetMsg(window->UserPo
633 pZ              rt);
634 uR              class=message->Class; code=message->Code;
635 8C              xx=(window->MouseX)-320; yy=(window->MouseY)-256;
636 7V              sprintf(string," X: %ld ",xx);
637 q2              Move(rport, 0, 8);
638 oy              Text(rport, (int)string, (int)(strlen(string)));
639 Dp              sprintf(string," Y: %ld ",yy);
640 t5              Move(rport, 290, 8);
641 Lq2              Text(rport, (int)string, (int)(strlen(string)));
642 CO              for(t=1; t<MA; t++)
643 9h              { x2[t]=(x[t]-(float)xx)*Faktor;
644 gP4                  y2[t]=(y[t]-(float)yy)*Faktor;
645 bX                  xalt[t]-=xx; yalt[t]-=yy; }
646 gK2              clearsreen(1);
647 Rw0      }
648 vw      void speichern()
649 Ps      {
650 Df2          lswindow.Title = " Save "; lsgadwart(); save();
651 V00      }
652 6l      void laden()
653 Tw      {
654 XV2          if((booltest(" Achtung! ",
655 Im5              "Wirklich laden? (J/N)")) == 1)
656 Wz2          {
657 cL4              lswindow.Title = " Load ";
658 CG              lsgadwart(); SetAPen(rport, 0);
659 RS              for(t=1; t<=anz; t++)
660 yZ6                  WritePixel(rport, xalt[t], yalt[t]);
661 Ti4              load(); SetAPen(rport, 2);
662 sz              if(start == 2)
663 B06                  mt[9].IText = "Fixstern (JA)";
664 XK4              else
665 ma6                  mt[9].IText = "Fixstern (NEIN)";
666 YZ4              for(t=1; t<=anz; t++)
667 Jo6                  WritePixel(rport, (xalt[t]=320+(int)(x[t]/Faktor)),
668 558                      (yalt[t]=256+(int)(y[t]/Faktor)));
669 nI2              }
670 oJO      }

```

Listing 1. (Fortsetzung)

SIMULATION

```

671 9X2 /* ----- */
672 yt1 /* ---- Hauptprogramm ---- */
673 BZ0 /* ----- */
674 Zv main()
675 p1 {
676 LM2  openall(); startwerte(); clearscreen(0); code=0; class=0;
677 st   for(t=1; t<MA; t++) wertaufbereiten();
678 PT   while(1)
679 tM   {
680 BX4   while(class != MENUPICK)
681 9E     { message = (struct IntuiMessage *)GetMsg(window->User
Port);
682 v26     code = message->Code; class=message->Class; }
683 x14   if (code == MENUNULL) { code = 0; class = 0; }
684 re     else
685 zS     {
686 8S6       class = 0;
687 04       switch( ITEMNUM (code) )
688 2V       {
689 vT8         case 0: /* Start */
690 YHA           code2 = code; SetAPen(rport, 2);
691 xy           for(t=1; t<anz; t++)
692 U5C             WritePixel(rport, xalt[t], yalt[t]);
693 B3A           bewegung();
694 QQ           if((code = message->Code)==MENUNULL)
695 C2C             code = code2;
696 9IA           break;
697 uT8         case 1: /* Stop */
698 G8A           code = 0; break;
699 ey8         case 2: /* Neues Objekt */
700 7CA           code = 0; neu(); break;
701 lw8         case 3: /* Masse */
702 a1A           code = 0; if(anz>0) masse(); break;
703 VZ8         case 4: /* Position */
704 zUA           code = 0; if(anz>0) position(); break;
705 AA8         case 5: /* Geschwindigkeit */
706 rfA           code = 0; if(anz>0) geschw(); break;
707 my8         case 6: /* Löschen */
708 x7A           code = 0; if(anz>0) loesch(); break;
709 T18         case 7: /* Zeigen */
710 ORA           code = 0; if(anz>0) zeig(); break;
711 Ev8         case 8: /* Zeitintervall */
712 JmA           code = 0; zeit(); break;
713 HL8         case 9: /* Fixstern (JA/NEIN) */
714 bKA           /
715 fl           code = 0;
716 rB           if(start == 1)
717 OB             { start=2; mt[9].IText = "Fixstern (JA)"; }
718 M9             else
719 Wf             { start=1; mt[9].IText = "Fixstern (NEIN)"; }
720 j48             break;
721 meA           case 10: /* Vergrößerung */
722 yr8             code = 0; vergr(); clearscreen(1); break;
723 frA           case 11: /* Uhr */
724 zo8             code = 0; uhr(); break;
725 I3A           case 12: /* Gravo - Grafik */
726 D18             code = 0; gragraf(); break;
727 itA           case 13: /* Neuer Nullpunkt */
728 aK8             code = 0; neunull(); break;
729 qZA           case 14: /* Bildschirm löschen */
730 lj           /
731 CFN             code = 0;
732 41C             if((booltest(" Achtung! ",
"Bildschirm löschen ? (J/N)")) == 1)
733 ktA               clearscreen(1);
734 Lc8             break;
735 hHA           case 15: /* Laden */
736 XK8             code = 0; laden(); break;
737 lNA           case 16: /* Speichern */
738 E68             code = 0; if(anz>0) speichern(); break;
739 OJA           case 17: /* Neu */
740 vt             code = 0;
741 hiN             if((booltest(" Achtung! ",
"Wirklich neu? (J/N)")) == 1)
742 3vA               { startwerte();
743 wxC                 for(t=1; t<MA; t++) wertaufbereiten();
744 ad                 clearscreen(); }
745 w5A             break;
746 GQ8           case 18: /* Ende */
747 8rA             code = 0;
748 31             if((booltest(" Achtung! ",
"Wirklich beenden? (J/N)")) == 1)
749 J2N               {
750 2VA                 ClearMenuStrip(window); CloseWindow(window);
751 2VC

```

```

752 C3           CloseScreen(screen); CloseLibrary(IntuitionBase
);
753 s8           CloseLibrary(GfxBase); exit(TRUE);
754 AFA         }
755 Bg6         }
756 Ch4         }
757 D12         }
758 EJO         }
(C) 1987 M&T

```

Listing 1. (Schluß)

Programmname: DeRev.IO.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Compiler: Aztek C V3.4

Aufrufe: cc DeRev.IO.c

Bemerkung: Dieses Programm bitte zuerst
compilieren, danach »DeRev.Main.c«

Programm : DeRev

```

1 RWO /*****
2 CO ** De revolutionibus orbium coelestium I/O **
3 DF ** ( DeRev.IO.c ) **
4 Dq ** (c) 1988 by **
5 Ju ** Jürgen Brendel **
6 HQ *****/
7 hQ #define MA 100
8 AE extern char t,anz,start,buffer[71];
9 wK extern float x[MA],y[MA],x2[MA],y2[MA],Vx[MA],Vy[MA],
10 l1E m[MA],gm[MA],Zeit,gam;
11 in0 extern long xalt[MA],yalt[MA];
12 zR void save()
13 9c {
14 Ol2 int handle,nix;
15 mz if((handle = creat(buffer, nix)) != -1)
16 Cf {
17 qs4 write(handle, &anz, sizeof(char));
18 ff write(handle,&start, sizeof(char));
19 YD write(handle, x, sizeof(x));
20 jQ write(handle, y, sizeof(y));
21 Ms write(handle, x2, sizeof(x2));
22 W4 write(handle, y2, sizeof(y2));
23 su write(handle, Vx, sizeof(Vx));
24 43 write(handle, Vy, sizeof(Vy));
25 sB write(handle, m, sizeof(m));
26 rm write(handle, &Zeit, sizeof(float));
27 6h close(handle);
28 Sx2 }
29 Ty0 }
30 IV void load()
31 Ru {
32 gJ2 int handle,nix;
33 3t if((handle = open(buffer, 01, nix)) != -1)
34 Ux {
35 VR4 read(handle, &anz, sizeof(char));
36 xG read(handle,&start, sizeof(char));
37 q9 read(handle, x, sizeof(x));
38 zK read(handle, y, sizeof(y));
39 4x read(handle, x2, sizeof(x2));
40 C7 read(handle, y2, sizeof(y2));
41 QT read(handle, Vx, sizeof(Vx));
42 af read(handle, Vy, sizeof(Vy));
43 WT read(handle, m, sizeof(m));
44 mS read(handle, &Zeit, sizeof(float));
45 UY close(handle); gam=.00667*(Zeit/1.0e5);
46 77 for(t=1; t<MA; t++) gm[t]=gam*m[t];
47 lG2 }
48 mHO }
(C) 1987 M&T

```

**Listing 2. »DeRev.IO.c« ist als erstes Listing zu
compilieren. Das Hauptprogramm benötigt diese Rou-
tine. Bitte mit dem Checksummer (Seite 159) eingeben.**

»Floppy-Speeder« mit Niveau

Der Amiga ist ein schneller Computer — solange er keine Directories von Disketten laden soll. Durch die komplizierte Organisation des Inhaltsverzeichnisses muß der Lesekopf des Disketten-Laufwerkes oft bewegt werden, was als »Sägen« nur zu gut hörbar ist. »FastLoadCopy« schafft Abhilfe und sortiert die Blöcke des Directory und aller Programme. Dadurch werden die Diskettenzugriffe wesentlich beschleunigt und das »Sägen« reduziert. Einzige Voraussetzung zum Arrangieren der Disketten ist 1 MByte Hauptspeicher.

Für die Lesegeschwindigkeit von Amiga-Diskettenlaufwerken gibt es eine einfache Gleichung: Je mehr sich der Schreib-/Lesekopf bewegt, desto niedriger ist die Lesegeschwindigkeit. Beim Laden eines Directory (oder dem Doppelklick auf ein Diskettensymbol) bekommt man das besonders stark zu spüren: Der Schreib-/Lesekopf bewegt sich hin und her, daß es eine wahre Freude ist. Entsprechend lang dauert es, bis der Computer

Haben Sie sich auch schon geärgert, wenn der Amiga beim Laden von Directories und Programmen von Diskette wie wild auf dieser »schrubbt«? Dies ist ein Zeichen dafür, daß die Filestruktur auf der Diskette mächtig durcheinander ist. »FastLoadCopy« beseitigt diesen Umstand und räumt auf Ihren Disketten ordentlich auf. Die Geschwindigkeit beim Laden wird dadurch beträchtlich erhöht. So ganz nebenbei ist unser Listing auch noch ein schnelles Kopierprogramm.

das Inhaltsverzeichnis eingelesen hat. Um das Ganze zu beschleunigen, müßte man also die Anzahl der Bewegungen reduzieren — genau das macht »FastLoadCopy«.

Kopie schlägt Original!

Dieses Programm funktioniert nach einem einfachen Prinzip: Zuerst einmal werden alle Blöcke der zu bearbeiten-

den Diskette in den Speicher gelesen. Daher ist es auch unbedingt nötig, daß Ihr Amiga mindestens 1 MByte Speicher besitzt — auf eine Amiga-Diskette passen ja 880 KByte. Stehen die Daten im Speicher, werden sie vom Programm nach einem bestimmten Schema sortiert. Dadurch wird die optimale Geschwindigkeit beim Lesen des Directory und Laden von Programmen erreicht. Nach dem Sortiervorgang — er dauert etwa eine

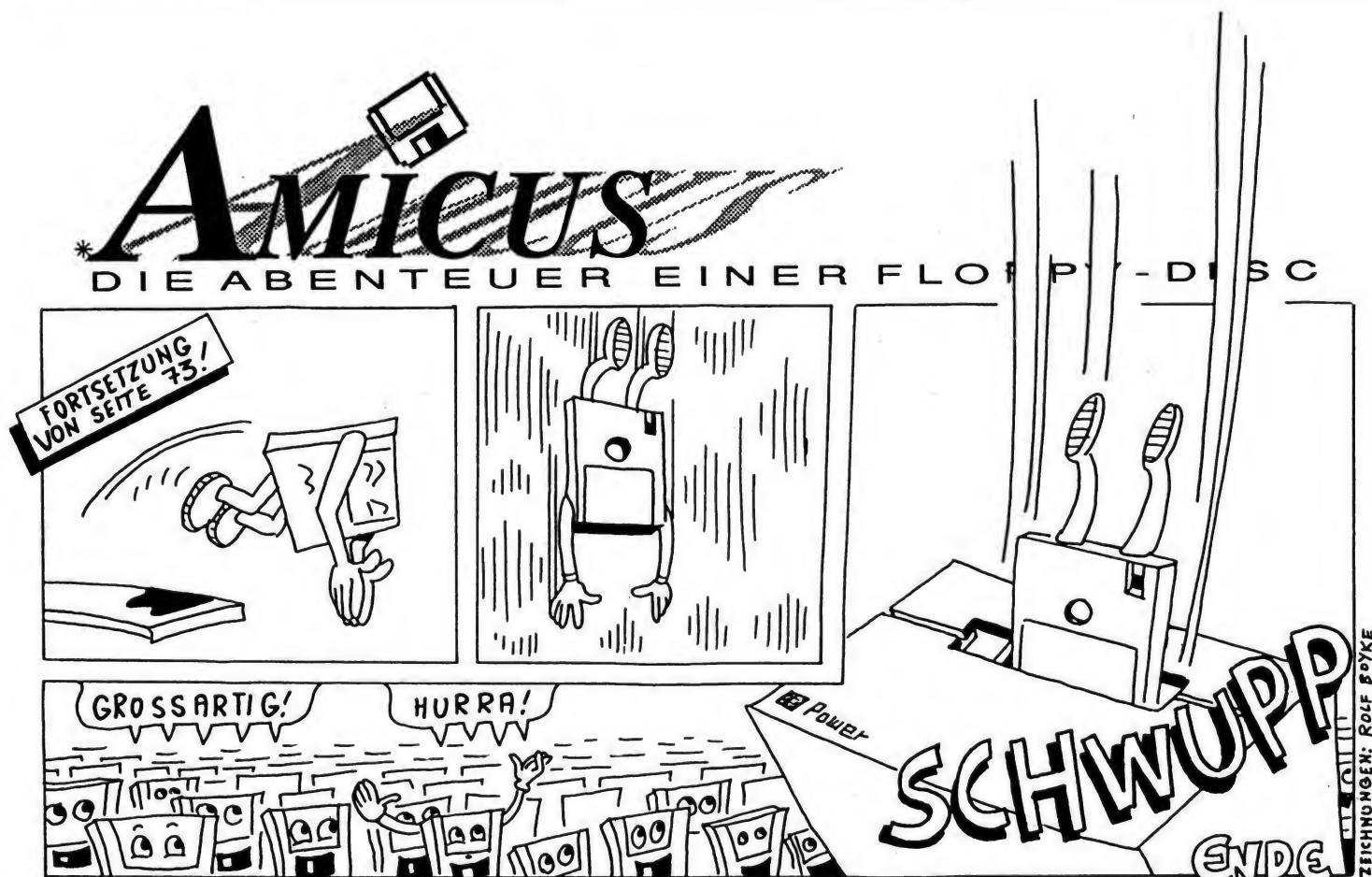
halbe Minute — werden die nun sortierten Blöcke wieder auf die Diskette zurückgeschrieben.

Vielleicht werden Sie nun fragen: »Wieso heißt das Programm dann FastLoadCopy und nicht einfach FastLoad?«. Nun, nachdem eine Diskette beschrieben worden ist, kann man wahlweise eine weitere Diskette mit denselben Daten (die ja noch im Speicher stehen) bespielen. Unser Listing läßt sich also auch als Kopierprogramm, das zudem noch schnell ist, verwenden.

Eingabehinweise

Tippen Sie bitte Listing 1 mit dem Checksummer (Seite 159) ab. »FastLoadCopy« ist in Assembler geschrieben. Damit auch Leser, die keinen Assembler besitzen, das Programm eingeben können, haben wir dieses als DATA-Lader abgedruckt. Dieser erzeugt das Programm »FLC« auf Diskette. Den Quellcode finden Sie in Listing 2.

Wenn Sie in Ihrem Computer nur 1 MByte Speicher haben,



beachten Sie bitte: Zusätzliche Diskettenlaufwerke und Festplatten sollten vor dem Start des Programmes entfernt werden, da diese zu viel Speicherplatz verbrauchen. Um das Programm zu starten, tippen Sie im CLI:

FLC

Legen Sie nun die Diskette, die Sie bearbeiten wollen, in das interne Diskettenlaufwerk des Amiga und drücken Sie die linke Maustaste. Sollten Sie nun die Meldung »Zu wenig Speicher« erhalten, so beachten Sie bitte die Hinweise in Tabelle 1. Ist genügend Speicher vorhanden, so lädt das Programm die Daten von der Diskette in den Computer (»...loading«). Ist das Laden beendet, so werden die Blöcke sortiert (»...changing«).

Dabei treten bei fast allen Disketten einige Fehlermeldungen auf, zum Beispiel »Fehler in Block \$032b. Disk u.U. fehlerhaft! Fehlernummer 1«. Wichtig ist in diesem Fall die Fehlernummer (siehe auch Tabelle 2). Bei der abgedruckten Version werden Fehlermeldungen mit Text ausgegeben. Auf der Diskette zu diesem Sonderheft befindet sich noch eine Version, welche kurze Kommentare ausgibt (sie trägt den Namen »FLCO«).

Trägt die Fehlermeldung die Nummer »1«, so hat das für den Ablauf des Programmes keine Bedeutung. Sie können in diesem Fall also getrost fortfahren. Höhere Fehlernummern als »1« müssen nicht unbedingt

Nummer	Kurzbeschreibung des Fehlers
1	Fehlendes "First Data" in FileListBlock
2	Falscher "Parent" - Eintrag
3	Datenblöcke nicht zusammenhängend
4	Falscher "Parent" in FileListBlock
5	Falsches "Extension" in FileListBlock
6	Erstblockeintrag fehlt in FileHeader
8	Falscher "HeaderKey" im Block HardError! Keine Wiederaufnahme des Sortierens möglich: beispielsweise kein DOS-Format, falsche Bitmap, Programmfehler

Fehlernummer 1 ist unbedenklich. Bei Fehlernummern größer als 1, sollte zuerst auf eine Versuchsdiskette gespeichert werden.
Bei Fehlernummer 8 bricht das Programm selbständig ab.

Tabelle 2. Die Fehlermeldungen und ihre Bedeutung

problematisch sein, man sollte die Daten aber vorsichtshalber erst einmal auf eine Versuchsdiskette speichern. Bei Fehlernummer »8« wird der Sortiervorgang abgebrochen, da es sich hierbei um einen Fehler handelt, der es unmöglich macht, das Programm fortzusetzen.

Sortieren ist alles

Sind die Daten schließlich sortiert, so werden Sie aufgefordert, die Zieldiskette einzulegen und die linke Maustaste zu drücken. Normalerweise sind Quell- und Zieldiskette natürlich identisch, da die Blöcke ja sortiert wieder auf die Queldiskette zurückgeschrieben werden sollen. Sie können aber natürlich Ihre Daten ge-

nausgut auf eine andere Diskette speichern und FastLoadCopy so als Kopierprogramm verwenden. Haben Sie beim Sortieren höhere Fehlernummern als 1 bekommen, verwenden Sie bitte ebenfalls eine andere Diskette als Zieldiskette, um Datenverluste zu vermeiden! Die Zieldiskette darf

natürlich nicht schreibgeschützt sein.

Nachdem die Daten auf die Zieldiskette geschrieben wurden (»...saving«), haben Sie die Wahl, mit Hilfe der linken Maustaste das Programm zu verlassen oder es mit der rechten Maustaste wieder zu starten. Haben Sie die rechte Maustaste gedrückt, so folgt die Frage

LMaustaste = Diskkopie

RMaustaste = Ramkopie

»Diskkopie« bedeutet, daß das Programm wieder von vorne abläuft; mit »Ramkopie« läßt sich eine weitere Diskette mit den Daten beschreiben, die noch im Speicher sind. Sie werden wiederum aufgefordert, die Zieldiskette einzulegen und die linke Maustaste zu drücken. FastLoadCopy ist also auch als Mehrfach-Kopierprogramm gut geeignet. Interessierte finden in Tabelle 3 eine kurze Beschreibung der Funktionsweise des Programms.

(Paul Sponagl/
A. Lietz/M. Jobst/rs)

1.	Meldung ausgeben und Speicher reservieren. (»getmem«)
2.	Mit Hilfe des »Trackdisk.device« werden die Tracks der Diskette geladen. (»losablock«)
3.	Taskswitching abschalten. (»taskoff«)
4.	Bitmap nach Block 879 verschieben
5.	Blöcke ordnen. Schema: RootBlock File-Header File-List User-Dir Datenblock ... Datenblöcke ... File-Header File-List Datenblock ... Datenblöcke ... und so weiter.
Die Hash-Tabelle des ersten User-Dirs beziehungsweise des Root-Blocks wird der Reihe nach durchsucht und deren File-Header oder User-Dirs hintenangestellt.	
Hinter die File-Header kommen noch die File-Listblöcke und — falls nur ein Datenblock im File-Header eingetragen ist — dieser selbst. (Beispiel: »info«-Datei für die Workbench). Darauf folgen die restlichen Datenblöcke des Directory. Dann sucht das Programm das nächste User-Dir und beginnt wieder von vorne. (»DiskSort« und »SeHashB«)	
6.	Diskette speichern (»Savedisk«)
7.	»Maustastenrequester« ausgeben (»ende«)
8.	Falls »Nochmal« und »Ramkopie« gewählt wurde: zurück nach »6«.
9.	Bei »Nochmal« und »Diskkopie«: zurück nach »1«.
10.	Bei »Quit« Speicher freigeben und »Ende« (»freemem«)

Tabelle 3. Eine Kurzübersicht des Programmes. Der Sourcecode von »FastLoadCopy« befindet sich auf der Programmservice-Diskette zu diesem Sonderheft.

Bei Computern mit 1 MByte Speicher

Da eine Diskette schon 880 K im Speicher belegt, ist nur noch sehr wenig freier Platz im Speicher — zu wenig, um zum Beispiel die Workbench, eine RAM-Disk oder den Speicherbereich aufzunehmen, der für externe Diskettenlaufwerke und Festplatten erforderlich ist. Auch die Benutzung von anderen Programmen vor dem Start von »FastLoadCopy« verbraucht meist zu viel kostbaren Speicherplatz.

Am besten kopieren Sie das Programm FastLoadCopy nach dem Abtippen in das Hauptverzeichnis oder das Verzeichnis »C« Ihrer Workbenchdiskette. Entfernen Sie sämtliche externen Laufwerke und lösen Sie mit <CTRL COMMODORE AMIGA> einen Reset aus.

Legen Sie Ihre Workbenchdiskette in das interne Laufwerk und warten Sie, bis die Einschaltmeldung auf dem Bildschirm erscheint. Dann drücken Sie sofort <CTRL D>, um die »startup-sequence« abzubrechen, die Ihr Computer nun normalerweise ausführen würde. Jetzt starten Sie FastLoadCopy. Nun sollten eigentlich keine Probleme mehr auftreten.

Bei Computern mit mehr als 1 MByte Speicher

Hier kann es eigentlich nur passieren, daß ein bereits geladenes Programm zuviel Speicherplatz verbraucht. Schalten Sie also laufende Programme ab oder benutzen Sie FastLoadCopy nach einem Reset Ihres Rechners.

Tabelle 1. So lösen Sie das Speicherplatzproblem

Programmname:	FLC_GEN
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic
Bemerkung:	Generiert das lauffähige Maschinenprogramm »FLC« auf Diskette im Laufwerk df0:

Programmname: FLC_GEN

```

1 OmD REM Generiert lauffähiges Programm
2 ag CLS
3 ss OPEN "df0:FLC" FOR OUTPUT AS 1
4 BS READ anz
5 oa FOR i=1 TO anz
6 3n1 READ h$
7 yB2 wert1=ASC(LEFT$(h$,1))
8 bP IF wert1>64 THEN wert1=wert1-87
    ELSE wert1=wert1-48
9 FI wert1=wert1*16
10 7c wert2=ASC(RIGHT$(h$,1))
11 wp IF wert2>64 THEN wert2=wert2-87
    ELSE wert2=wert2-48
12 P1 wert=wert1+wert2
13 9G PRINT #1,CHR$(wert);
14 J00 NEXT
15 3n CLOSE 1
16 Ov END
17 yc Werte:
18 FR DATA 5128
19 ph DATA 00,00,03,f3,00,00,00,00,00,00
20 Qc DATA 00,01,00,00,00,00,00,00,00,00
21 A0 DATA 00,00,03,d5,00,00,03,e9,00,00
22 vw DATA 03,d5,23,cf,00,00,09,36,42,39
23 4r DATA 00,00,09,34,23,fc,00,00,0d,5c
24 qN DATA 00,00,0f,4a,23,fc,00,00,00,42
25 lN DATA 00,00,0f,4e,4e,b9,00,00,0c,f4
26 38 DATA 23,fc,00,00,0d,ba,00,00,0f,4a
27 lG DATA 23,fc,00,00,00,1c,00,00,0f,4e
28 YJ DATA 4e,b9,00,00,0c,f4,4e,b9,00,00
29 aC DATA 0b,34,23,fc,00,00,00,03,00,00
30 Mw DATA 0b,90,23,fc,00,06,e0,00,00,00
31 Kz DATA 0b,98,4e,b9,00,00,0b,5a,4a,b9
32 KV DATA 00,00,0b,94,67,00,09,04,23,f9
33 sf DATA 00,00,0b,94,00,00,0b,2c,23,fc
34 2Q DATA 00,00,00,05,00,00,0b,90,4e,b9
35 dz DATA 00,00,0b,5a,4a,b9,00,00,0b,94
36 p7 DATA 67,00,08,e0,23,f9,00,00,0b,94
37 l5 DATA 00,00,0b,30,23,fc,00,00,0e,3c
38 q7 DATA 00,00,0f,4a,23,fc,00,00,0d,0d
39 Fb DATA 00,00,0f,4e,4e,b9,00,00,0c,f4
40 lP DATA 23,f9,00,00,0b,98,00,00,0c,e6
41 gR DATA 23,f9,00,00,0b,2c,00,00,0c,ea
42 q2 DATA 33,fc,00,02,00,00,0c,f2,23,fc
43 XM DATA 00,00,03,70,00,00,0c,ee,4e,b9
44 ge DATA 00,00,0b,9c,bc,b9,00,00,0b,98
45 8B DATA 66,00,08,a6,4e,b9,00,00,0b,0e
46 a2 DATA 23,fc,00,00,00,00,00,00,0c,ee
47 l1 DATA 4e,b9,00,00,0b,9c,bc,b9,00,00
48 vh DATA 0b,98,66,00,08,86,23,fc,00,00
49 lC DATA 0e,49,00,00,0f,4a,23,fc,00,00
50 Rf DATA 00,0e,00,00,0f,4e,4e,b9,00,00
51 BJ DATA 0c,f4,2c,79,00,00,00,04,4e,ae
52 8T DATA ff,7c,20,3c,00,00,1b,90,42,81
53 gN DATA 4e,ae,ff,3a,4a,80,67,00,08,30
54 pz DATA 23,c0,00,00,07,ba,20,79,00,00
55 kS DATA 0b,30,20,28,01,3c,33,00,00,00
56 kc DATA 07,b6,21,7c,00,00,03,6f,01,3c
57 Yt DATA 4e,b9,00,00,05,04,23,c8,00,00
58 lJ DATA 05,00,22,3c,00,00,03,6f,74,01
59 OQ DATA 4e,b9,00,00,01,fa,23,f9,00,00
60 rN DATA 07,ae,00,00,05,00,33,c1,00,00
61 6S DATA 07,b6,2e,3c,00,00,03,71,28,07
62 QA DATA 20,3c,00,00,03,70,42,86,7a,02
63 Zr DATA 33,c7,00,00,07,4a,4e,b9,00,00
64 mJ DATA 05,36,20,04,52,84,b8,bc,00,00
65 oJ DATA 06,e0,66,02,78,02,4e,b9,00,00
66 h7 DATA 04,d6,66,20,b8,bc,00,00,03,6f
67 Lq DATA 67,18,4e,b9,00,00,05,04,24,10
68 sz DATA b4,bc,00,00,00,08,67,d2,94,a8
69 XV DATA 01,fc,67,bc,60,ca,4e,b9,00,00
70 v9 DATA 07,50,2c,79,00,00,00,04,22,79

```

```

71 Oj DATA 00,00,07,ba,20,3c,00,00,1b,90
72 Q4 DATA 4e,ae,ff,2e,60,00,08,94,48,e7
73 gZ DATA fe,fe,4e,b9,00,00,05,04,23,c8
74 zx DATA 00,00,07,ae,23,c0,00,00,07,a6
75 Qo DATA 23,c1,00,00,07,aa,20,01,4e,b9
76 h5 DATA 00,00,05,04,23,c8,00,00,07,b2
77 lJ DATA 4e,b9,00,00,02,a6,20,39,00,00
78 9n DATA 07,a6,23,f9,00,00,07,aa,00,00
79 hS DATA 07,a6,23,c0,00,00,07,aa,20,39
80 MT DATA 00,00,07,ae,23,f9,00,00,07,b2
81 q1 DATA 00,00,07,ae,23,c0,00,00,07,b2
82 LF DATA 42,79,00,00,07,b8,20,7c,00,00
83 H1 DATA 00,00,4e,b9,00,00,08,3a,20,39
84 IT DATA 00,00,07,a6,4e,b9,00,00,04,d6
85 dS DATA 66,18,20,79,00,00,07,ae,24,10
86 PD DATA b4,3c,00,08,67,04,94,a8,01,fc
87 UP DATA 4e,b9,00,00,02,a6,4e,b9,00,00
88 E1 DATA 08,68,4e,b9,00,00,07,e2,4e,b9
89 38 DATA 00,00,07,c0,4c,7f,7f,4e,75
90 ns DATA 2c,39,00,00,07,a6,b4,3c,00,05
91 gG DATA 67,2c,b4,3c,00,08,67,00,01,14
92 Bq DATA 4a,02,67,00,00,e8,b4,3c,00,13
93 sx DATA 67,00,00,9e,b4,3c,00,01,67,0e
94 Zg DATA 13,fc,00,08,00,00,09,34,4e,b9
95 rp DATA 00,00,08,b4,4e,75,48,e7,ff,fe
96 iy DATA 4e,b9,00,00,04,ac,42,85,20,28
97 Hv DATA 01,f4,4e,b9,00,00,05,36,22,48
98 kV DATA 20,28,00,10,4e,b9,00,00,05,04
99 ZQ DATA bc,a8,00,04,67,0e,13,fc,00,03
100 09 DATA 00,00,09,34,4e,b9,00,00,08,b4
101 xz DATA 33,fc,00,04,00,00,07,b8,4e,b9
102 90 DATA 00,00,08,3a,20,28,00,10,4a,80
103 PP DATA 66,d0,20,49,20,28,01,f8,4a,80
104 tK DATA 67,2a,4e,b9,00,00,05,04,bc,a8
105 lU DATA 01,f4,67,0e,13,fc,00,04,00,00
106 xt DATA 09,34,4e,b9,00,00,08,b4,33,fc
107 X4 DATA 01,f4,00,00,07,b8,4e,b9,00,00
108 YF DATA 08,3a,60,ce,4c,df,7f,ff,4e,75
109 F1 DATA 48,e7,ff,fe,4e,b9,00,00,04,ac
110 Gf DATA 20,28,01,f4,4e,b9,00,00,05,04
111 5m DATA 20,28,01,f8,4a,80,66,0e,13,fc
112 lY DATA 00,05,00,00,09,34,4e,b9,00,00
113 lX DATA 08,b4,bc,80,66,e0,33,fc,01,f8
114 NU DATA 00,00,07,b8,4e,b9,00,00,08,3a
115 V7 DATA 4c,df,7f,ff,4e,75,48,e7,ff,fe
116 Sg DATA 4e,b9,00,00,04,ac,20,28,01,f4
117 LW DATA 42,85,4e,b9,00,00,05,36,7a,01
118 3t DATA 20,06,4e,b9,00,00,05,36,4c,df
119 lJ DATA 7f,ff,4e,75,48,e7,ff,fe,20,79
120 16 DATA 00,00,07,ae,20,28,00,04,22,28
121 f3 DATA 00,08,b2,7c,00,01,67,68,4e,b9
122 ms DATA 00,00,05,04,d1,fc,00,00,00,18
123 gh DATA 72,47,bc,98,57,c9,ff,fc,67,0e
124 2E DATA 20,79,00,00,05,32,20,28,01,f8
125 To DATA 24,48,60,dc,33,fc,ff,fc,00,00
126 Y3 DATA 07,b8,4e,b9,00,00,08,3a,4a,90
127 eD DATA 67,04,20,10,60,04,20,2a,00,18
128 BB DATA 4e,b9,00,00,05,04,bc,a8,00,10
129 x1 DATA 67,0e,13,fc,00,06,00,00,09,34
130 ce DATA 4e,b9,00,00,08,b4,33,fc,00,10
131 e1 DATA 00,00,07,b8,4e,b9,00,00,08,3a
132 Fe DATA 60,5a,4e,b9,00,00,05,04,bc,a8
133 BA DATA 01,34,67,0e,13,fc,00,06,00,00
134 PL DATA 09,34,4e,b9,00,00,00,08,b4,33,fc
135 sa DATA 01,34,00,00,07,b8,4e,b9,00,00
136 gk DATA 08,3a,bc,a8,00,10,67,0e,13,fc
137 mp DATA 00,01,00,00,09,34,4e,b9,00,00
138 No DATA 08,b4,33,fc,00,10,00,00,07,b8
139 tT DATA 4e,b9,00,00,08,3a,20,28,01,f8
140 T4 DATA 4a,80,67,08,4e,b9,00,00,05,04
141 cT DATA 60,ce,4c,df,7f,ff,4e,75,20,79
142 q6 DATA 00,00,07,ae,bc,a8,00,04,67,0e
143 KQ DATA 13,fc,00,07,00,00,09,34,4e,b9
144 lN DATA 00,00,08,b4,33,fc,00,04,00,00
145 In DATA 07,b8,4e,b9,00,00,08,3a,4e,75
146 z4 DATA 48,e7,80,00,2a,39,00,00,05,00

```

```

147 Wd DATA 72,02,58,85,06,41,00,20,b0,41
148 Eh DATA 6a,f6,92,40,cb,88,2a,10,04,41
149 G4 DATA 00,20,44,41,03,05,4c,df,00,01
150 31 DATA 4e,75,00,00,00,00,48,e7,80,00
151 E1 DATA b0,7c,03,70,6a,08,20,79,00,00
152 RH DATA 0b,2c,60,0a,20,79,00,00,0b,30
153 u1 DATA 04,40,03,70,c0,fc,02,00,d1,c0
154 Wp DATA 23,c8,00,00,05,32,4c,df,00,01
155 85 DATA 4e,75,00,00,00,00,48,e7,f8,c0
156 uv DATA 4e,b9,00,00,05,04,d1,fc,00,00
157 RO DATA 00,18,76,47,20,18,4a,80,67,48
158 6p DATA 4e,b9,00,00,05,d8,ba,bc,00,00
159 Eu DATA 00,02,66,08,20,39,00,00,07,a6
160 2E DATA 60,04,bc,80,67,44,22,48,4e,b9
161 pt DATA 00,00,05,04,20,28,01,f0,4a,80
162 B3 DATA 67,1c,4e,b9,00,00,05,d8,ba,bc
163 qg DATA 00,00,00,02,66,08,20,39,00,00
164 l5 DATA 07,a6,60,dc,bc,80,67,20,60,d6
165 OH DATA 20,49,51,cb,ff,b0,4a,85,66,26
166 jq DATA 13,fc,00,08,00,00,09,34,4e,b9
167 tv DATA 00,00,08,b4,91,fc,00,00,01,f4
168 lB DATA 4a,85,66,e8,33,fc,01,f0,00,00
169 ST DATA 07,b8,4e,b9,00,00,08,3a,bc,bc
170 Qb DATA 00,00,00,02,66,06,4e,b9,00,00
171 M1 DATA 06,bc,4c,df,03,1f,4e,75,4a,85
172 oE DATA 67,3a,ba,bc,00,00,02,67,34
173 Uy DATA 48,e7,80,80,4e,b9,00,00,05,04
174 CX DATA 20,28,01,f4,bc,80,67,0e,13,fc
175 UT DATA 00,02,00,00,09,34,4e,b9,00,00
176 yL DATA 08,b4,33,fc,01,f4,00,00,07,b8
177 wp DATA 4e,b9,00,00,08,3a,4c,df,01,01
178 oF DATA 4e,75,48,e7,e0,80,4e,b9,00,00
179 JV DATA 05,04,24,10,b4,3c,00,08,67,04
180 80 DATA 94,a8,01,fc,22,07,4e,b9,00,00
181 vY DATA 01,fa,52,87,52,86,4e,b9,00,00
182 MJ DATA 07,3e,b4,bc,00,00,00,05,66,70
183 Q3 DATA 23,f9,00,00,07,b8,00,00,07,4c
184 20 DATA 20,39,00,00,07,a6,4e,b9,00,00
185 eL DATA 05,04,20,28,01,f8,4a,80,67,1a
186 TD DATA 24,3c,00,00,00,13,22,07,4e,b9
187 O3 DATA 00,00,01,fa,52,87,52,86,4e,b9
188 oI DATA 00,00,07,3e,60,d2,b4,3c,00,13
189 Cm DATA 67,28,20,28,00,08,b0,bc,00,00
190 2P DATA 00,01,66,1e,20,28,00,10,22,07
191 MC DATA 24,3c,00,00,00,08,4e,b9,00,00
192 GJ DATA 01,fa,52,87,52,86,4e,b9,00,00
193 ny DATA 07,3e,23,f9,00,00,07,4c,00,00
194 09 DATA 07,a6,4c,df,01,07,4e,75,30,39
195 uW DATA 00,00,07,4a,48,e0,4a,86,67,3a
196 Ou DATA 4e,b9,00,00,05,04,22,10,b2,bc
197 QX DATA 00,00,00,08,67,1e,92,a8,01,fc
198 Ou DATA b2,bc,00,00,00,05,66,12,22,28
199 NO DATA 00,08,b2,bc,00,00,00,01,67,06
200 v1 DATA 4e,b9,00,00,07,0a,52,80,b0,bc
201 UT DATA 00,00,06,df,63,02,7e,02,51,ce
202 Nn DATA ff,e0,4e,75,48,e7,e0,00,20,28
203 2H DATA 00,10,67,24,22,07,24,3c,00,00
204 JR DATA 00,08,4e,b9,00,00,01,fa,52,87
205 lF DATA 4e,b9,00,00,07,3e,20,39,00,00
206 60 DATA 07,a6,4e,b9,00,00,05,04,60,d6
207 SC DATA 4c,df,00,07,4e,75,bc,bc,00,00
208 YH DATA 06,df,63,02,7e,02,4e,75,00,00
209 g7 DATA 00,00,00,00,70,01,52,40,b0,79
210 dd DATA 00,00,07,b6,67,2c,b0,7c,06,e0
211 Mo DATA 67,24,4e,b9,00,00,04,d6,66,e8
212 S1 DATA 4e,b9,00,00,05,04,42,84,76,7f
213 cW DATA d8,98,51,cb,ff,fc,98,a8,fe,14
214 f7 DATA 44,84,21,44,fe,14,60,cc,4e,75
215 Ep DATA 4e,b9,00,00,05,04,76,7e,d1,fc
216 Fm DATA 00,00,00,04,42,84,d8,98,51,cb
217 NU DATA ff,fc,44,84,21,44,fe,00,60,ac
218 Wn DATA 00,00,00,00,00,00,00,00,00,00
219 Xn DATA 00,00,00,00,00,00,00,00,00,00
220 tC DATA 00,00,00,00,00,00,48,e7,c0,c0
221 3n DATA 20,79,00,00,07,ae,22,79,00,00
222 bt DATA 07,b2,70,7f,22,10,20,d1,22,e1
223 Nf DATA 51,c8,ff,f8,4c,df,03,03,4e,75
224 al DATA 48,e7,fe,fe,20,39,00,00,07,a6
225 rD DATA 4e,b9,00,00,04,d6,2c,48,28,01
226 b3 DATA 2c,05,20,39,00,00,07,aa,4e,b9
227 Gt DATA 00,00,04,d6,bc,85,67,1c,09,06

```

Listing 1. Der Basic-Lader von »Fast LoadCopy«

228 Pa DATA 67,08,03,c5,67,08,09,c6,60,06
229 wB DATA 03,85,60,f6,09,86,2c,86,20,85
230 g4 DATA 4c,df,7f,7f,4e,75,09,06,67,08
231 Co DATA 03,c6,67,08,09,c6,60,06,03,86
232 9e DATA 60,f6,09,86,2c,86,60,e4,48,e7
233 Pn DATA 80,c0,30,39,00,00,07,b8,48,c0
234 nd DATA d1,c0,30,39,00,00,07,be,48,c0
235 JV DATA 22,79,00,00,07,ba,d3,c0,22,88
236 XQ DATA 58,80,33,c0,00,00,07,be,4c,df
237 SM DATA 03,01,4e,75,36,39,00,00,07,be
238 1x DATA e4,4b,53,43,20,79,00,00,07,ba
239 Ob DATA 22,39,00,00,07,a6,4a,90,67,20
240 vU DATA 22,58,22,81,51,cb,ff,f6,42,79
241 55 DATA 00,00,07,be,20,79,00,00,07,ba
242 vv DATA 36,3c,06,e3,42,98,51,cb,ff,fc
243 Ys DATA 4e,75,21,fc,00,00,00,04,22,39
244 4P DATA 00,00,07,aa,51,cb,ff,ce,60,d6
245 zB DATA 48,e7,ff,fe,20,39,00,00,07,a6
246 eK DATA 41,f9,00,00,0f,1e,76,02,4e,b9
247 b1 DATA 00,00,09,3a,41,f9,00,00,0f,48
248 rU DATA 10,39,00,00,09,34,48,80,42,83
249 pD DATA 4e,b9,00,00,09,3a,2c,79,00,00
250 CI DATA 00,04,4e,ae,ff,76,23,fc,00,00
251 uw DATA 0f,0d,00,00,0f,4a,23,fc,00,00
252 qB DATA 00,3d,00,00,0f,4e,4b,09,00,00
253 LB DATA 0c,f4,10,39,00,00,09,34,b0,3c
254 gJ DATA 00,07,63,16,4e,b9,00,00,07,50
255 WS DATA 4e,b9,00,00,0b,0e,2e,79,00,00
256 2v DATA 09,36,60,00,f8,ba,42,39,00,00
257 ku DATA 09,34,4c,df,7f,ff,4e,75,00,00
258 SP DATA 00,00,00,00,48,e7,f0,80,d1,c3
259 V7 DATA d1,fc,00,00,00,01,42,42,34,00
260 w3 DATA 02,42,00,0f,06,02,00,30,b4,7c
261 ZM DATA 00,39,63,00,00,06,06,42,00,07
262 10 DATA 11,02,e8,48,51,cb,ff,e2,4c,df
263 52 DATA 01,0f,4e,75,23,fc,00,00,0d,9e
264 Tk DATA 00,00,0f,4a,23,fc,00,00,0c,1c
265 tF DATA 00,00,0f,4e,4e,b9,00,00,0c,f4
266 1f DATA 4e,f9,00,00,0a,4a,23,fc,00,00
267 Yg DATA 0e,14,00,00,0f,0a,23,fc,00,00
268 qD DATA 00,1c,00,00,0f,4e,4e,b9,00,00
269 fF DATA 0c,f4,4e,f9,00,00,0a,4a,23,fc
270 xQ DATA 00,00,0e,57,00,00,0f,4a,23,fc
271 O8 DATA 00,00,00,5c,00,00,0f,4e,4e,b9
272 uG DATA 00,00,0c,f4,42,00,4e,b9,00,00
273 vd DATA 0a,2c,4a,00,67,00,00,78,23,fc
274 FA DATA 00,00,0e,b3,00,00,0f,4a,23,fc
275 06 DATA 00,00,00,5a,00,00,0f,4e,4e,b9
276 9d DATA 00,00,0c,f4,08,39,00,0a,00,df
277 3D DATA f0,16,67,00,ff,f6,42,00,4e,b9
278 GP DATA 00,00,0a,2c,4a,00,67,00,00,7a
279 mA DATA 23,fc,00,00,0d,5c,00,00,0f,4a
280 WK DATA 23,fc,00,00,00,42,00,00,0f,4e
281 d0 DATA 4e,b9,00,00,0c,f4,4e,b9,00,00
282 tc DATA 0b,0e,60,00,00,76,08,39,00,06
283 j1 DATA 00,bf,e0,01,67,00,00,12,08,39
284 Je DATA 00,0a,00,df,f0,16,66,00,ff,ea
285 zK DATA 06,00,00,01,4e,75,4a,b9,00,00
286 n1 DATA 0b,30,67,00,00,12,23,f9,00,00
287 Gz DATA 0b,30,00,00,0b,94,4e,b9,00,00
288 v1 DATA 0b,78,4a,b9,00,00,0b,2c,67,00
289 yU DATA 00,12,23,f9,00,00,0b,2c,00,00
290 bC DATA 0b,94,4e,b9,00,00,0b,78,4e,75
291 06 DATA 4e,b9,00,00,0a,4a,4e,f9,00,00
292 ry DATA 00,00,2c,79,00,00,00,04,4e,ae
293 Dz DATA ff,76,4a,39,00,00,09,34,66,00
294 Ku DATA ff,0e,23,fc,00,00,0d,d6,00,00
295 wE DATA 0f,4a,23,fc,00,00,00,1c,00,00
296 RF DATA 0f,4e,4e,b9,00,00,0c,f4,4e,b9
297 vX DATA 00,00,0b,34,23,fc,00,00,0e,30
298 xD DATA 00,00,0f,4a,23,fc,00,00,00,0c
299 Rn DATA 00,00,0f,4e,4e,b9,00,00,0c,f4
300 YW DATA 33,fc,00,0b,00,00,0c,f2,23,fc
301 p3 DATA 00,00,00,00,00,00,0c,ee,4e,b9
302 5v DATA 00,00,0b,9c,4e,b9,00,00,0b,0e
303 Co DATA 23,fc,00,00,03,70,00,00,0c,ee
304 wM DATA 4e,b9,00,00,0b,9c,4e,f9,00,00
305 2S DATA 09,ac,22,79,00,00,0b,2c,20,79
306 JO DATA 00,00,0b,30,20,3c,00,01,b8,00
307 An DATA 22,10,20,d1,22,c1,53,80,66,f6
308 Ig DATA 4e,75,00,00,00,00,00,00,00
309 Tr DATA 23,fc,00,00,0d,f2,00,00,0f,4a
310 w1 DATA 23,fc,00,00,00,22,00,00,0f,4e
311 ea DATA 4e,b9,00,00,0c,f4,08,39,00,06

312 vK DATA 00,bf,e0,01,66,f6,4e,75,2c,79
313 gJ DATA 00,00,00,04,20,39,00,00,0b,98
314 1H DATA 22,39,00,00,0b,90,4e,ae,ff,3a
315 j1 DATA 23,c0,00,00,0b,94,4e,75,2c,79
316 JM DATA 00,00,00,04,20,39,00,00,0b,98
317 qH DATA 22,79,00,00,0b,94,4e,ae,ff,2e
318 sq DATA 4e,75,00,00,00,00,00,00,00
319 MZ DATA 00,00,00,00,4d,f9,00,00,0c,76
320 rA DATA 20,3c,00,00,00,1b,2c,fc,00,00
321 fb DATA 00,00,51,c8,ff,f8,2c,79,00,00
322 1W DATA 00,04,20,39,00,00,0c,ee,c0,fc
323 RN DATA 02,00,23,c0,00,00,0c,ee,93,c9
324 KK DATA 4e,ae,fe,da,43,f9,00,00,0c,0c
325 ak DATA 23,40,00,10,4e,ae,fe,9e,43,f9
326 7k DATA 00,00,0c,76,20,3c,00,00,00,00
327 2J DATA 42,81,41,f9,00,00,0c,64,4e,ae
328 ST DATA fe,44,4a,80,66,00,00,6a,43,f9
329 xR DATA 00,00,0c,76,23,7c,00,00,0c,0c
330 ch DATA 00,0e,33,79,00,00,0c,f2,00,1c
331 Gw DATA 23,79,00,00,0c,ea,00,28,23,79
332 bY DATA 00,00,0c,e6,00,24,23,79,00,00
333 Jh DATA 0c,ee,00,2c,2c,79,00,00,00,04
334 ON DATA 4e,ae,fe,38,43,f9,00,00,0c,76
335 3V DATA 2c,29,00,20,33,7c,00,09,00,1c
336 1W DATA 23,7c,00,00,00,00,00,24,4e,ae
337 z9 DATA fe,38,43,f9,00,00,0c,0c,4e,ae
338 Mu DATA fe,98,43,f9,00,00,0c,76,4e,ae
339 fu DATA fe,3e,4e,75,74,72,61,63,6b,64
340 1D DATA 69,73,6b,2e,64,65,76,69,63,65
341 V1 DATA 00,00,00,00,00,00,00,00,00
342 Wm DATA 00,00,00,00,00,00,00,00,00
343 Xn DATA 00,00,00,00,00,00,00,00,00
344 Yo DATA 00,00,00,00,00,00,00,00,00
345 Zp DATA 00,00,00,00,00,00,00,00,00
346 aq DATA 00,00,00,00,00,00,00,00,00
347 br DATA 00,00,00,00,00,00,00,00,00
348 cs DATA 00,00,00,00,00,00,00,00,00
349 dt DATA 00,00,00,00,00,00,00,00,00
350 eu DATA 00,00,00,00,00,00,00,00,00
351 fv DATA 00,00,00,00,00,00,00,00,00
352 gw DATA 00,00,00,00,00,00,00,00,00
353 KQ DATA 00,00,00,00,00,00,00,00,48,e7
354 of DATA fe,fe,2c,79,00,00,00,04,43,f9
355 OM DATA 00,00,0d,4c,42,80,4e,ae,fe,68
356 op DATA 67,00,00,3a,23,c0,00,00,0d,58
357 xP DATA 2c,40,4e,ae,ff,04,22,00,24,39
358 eQ DATA 00,00,0f,4a,26,39,00,00,0f,4e
359 sp DATA 4e,ae,ff,d0,4a,80,6a,00,00,06
360 uz DATA 4e,ee,ff,70,2c,79,00,00,00,04
361 ta DATA 22,79,00,00,0d,58,4e,ae,fe,62
362 vE DATA 4c,df,7f,7f,4e,75,64,6f,73,2e
363 GS DATA 6c,69,62,72,61,72,79,00,00,00
364 D4 DATA 00,00,0c,0a,0a,09,09,31,3b
365 ux DATA 33,31,3b,34,32,6d,20,20,20,20
366 mo DATA 46,61,73,74,4c,6f,61,64,43,6f
367 Tv DATA 70,79,20,62,79,20,50,2e,53,2e
368 S9 DATA 20,20,20,28,43,29,20,4d,26,54
369 5a DATA 20,69,6e,20,31,39,38,20,20,20
370 v1 DATA 20,20,9b,30,6d,0a,0a,0a,5a,75
371 uJ DATA 20,77,65,6e,69,67,20,53,70,65
372 KK DATA 69,63,68,65,72,20,76,6f,72,68
373 Am DATA 61,6e,64,65,6e,0a,42,69,74,74
374 1w DATA 65,20,51,75,65,6c,6c,64,69,73
375 J8 DATA 6b,65,74,74,65,20,69,6e,20,44
376 Lo DATA 46,30,3a,0a,0a,42,69,74,74,65
377 FA DATA 20,5a,69,65,6c,64,69,73,6b,65
378 X7 DATA 74,74,65,20,69,6e,20,44,46,30
379 oJ DATA 3a,0a,2e,2e,2e,20,75,6e,64,20
380 92 DATA 6c,69,6e,6b,65,20,43,61,75,73
381 JY DATA 74,61,73,74,65,20,64,72,75,65
382 oJ DATA 63,6b,65,6e,21,0a,54,72,61,63
383 pM DATA 6b,64,69,73,6b,2e,64,65,76,69
384 Rz DATA 63,65,20,2d,20,50,72,6f,62,6c
385 1I DATA 65,6d,65,0a,2e,2e,2e,20,73,61
386 mH DATA 76,69,6e,67,0a,0a,2e,2e,2e,20
387 rk DATA 6c,6f,61,64,69,6e,67,0a,0a,2e
388 7V DATA 2e,2e,20,63,68,61,6e,67,69,6e
389 FA DATA 67,0a,0a,0a,0a,0a,09,9b,30,3b
390 T5 DATA 33,31,3b,34,32,6d,20,20,4c,4d
391 vv DATA 61,75,73,74,61,73,74,65,20,3d
392 1t DATA 20,20,51,75,69,74,20,20,20
393 9t DATA 20,9b,30,6d,20,20,20,20,20,20
394 PL DATA 20,20,20,20,20,20,20,9b,30,3b
395 4p DATA 33,31,3b,34,32,6d,20,20,52,4d

396 00 DATA 61,75,73,74,61,73,74,65,20,3d
397 0Q DATA 20,20,4e,6f,63,68,6d,61,6c,20
398 p9 DATA 20,9b,30,6d,0a,0a,09,9b,30,3b
399 eE DATA 33,31,3b,34,32,6d,20,20,4c,4d
400 44 DATA 61,75,73,74,61,73,74,65,20,3d
401 d2 DATA 20,44,69,73,6b,6b,6f,70,69,65
402 I2 DATA 20,9b,30,6d,20,20,20,20,20,20
403 YU DATA 20,20,20,20,20,20,20,9b,30,3b
404 Dy DATA 33,31,3b,34,32,6d,20,20,52,4d
405 99 DATA 61,75,73,74,61,73,74,65,20,3d
406 M7 DATA 20,52,61,6d,6b,6f,70,69,65,20
407 12 DATA 20,9b,30,6d,0a,46,65,68,6c,65
408 sU DATA 72,20,69,6e,20,42,6c,6f,63,6b
409 RP DATA 20,24,20,20,20,2e,20,44,69,73
410 vE DATA 6b,20,75,2e,55,2e,20,66,65,68
411 0Y DATA 6c,65,72,68,61,66,74,20,21,21
412 tN DATA 20,46,65,68,6c,65,72,6e,75,6d
413 VJ DATA 6d,65,72,20,20,0a,00,00,00,00
414 JI DATA 00,00,00,00,00,00,00,00,03,ec
415 x6 DATA 00,00,01,20,00,00,00,00,00,00
416 Zg DATA 00,02,00,00,00,08,00,00,00,0e
417 EI DATA 00,00,00,12,00,00,00,1c,00,00
418 ny DATA 00,22,00,00,00,28,00,00,00,2c
419 g1 DATA 00,00,00,36,00,00,00,3c,00,00
420 hx DATA 00,42,00,00,00,4c,00,00,00,56
421 LE DATA 00,00,00,5c,00,00,00,62,00,00
422 rw DATA 00,6c,00,00,00,70,00,00,00,7a
423 NL DATA 00,00,00,80,00,00,00,86,00,00
424 a8 DATA 00,90,00,00,00,94,00,00,00,9a
425 Km DATA 00,00,00,9e,00,00,00,a8,00,00
426 d4 DATA 0a,ae,00,00,00,b4,00,00,00,b8
427 2m DATA 00,00,00,be,00,00,00,c2,00,00
428 w0 DATA 0c,ca,00,00,d4,00,00,00,da
429 mG DATA 00,00,00,e0,00,00,00,ea,00,00
430 pN DATA 00,f4,00,00,00,fa,00,00,01,00
431 7y DATA 00,00,01,0a,00,00,01,0e,00,00
432 qa DATA 01,18,00,00,01,1e,00,00,01,40
433 y6 DATA 00,00,01,46,00,00,01,50,00,00
434 hy DATA 01,5e,00,00,01,64,00,00,01,72
435 kZ DATA 00,00,01,78,00,00,01,7c,00,00
436 rt DATA 01,82,00,00,01,9a,00,00,01,a0
437 uq DATA 00,00,01,b4,00,00,01,c4,00,00
438 UL DATA 01,dc,00,00,01,e8,00,00,02,00
439 ty DATA 00,00,02,06,00,00,02,0c,00,00
440 EJ DATA 02,12,00,00,02,1a,00,00,02,20
441 DC DATA 00,00,02,26,00,00,02,2c,00,00
442 as DATA 02,32,00,00,02,36,00,00,02,3c
443 d1 DATA 00,00,02,42,00,00,02,48,00,00
444 yQ DATA 02,4c,00,00,02,52,00,00,02,58
445 f0 DATA 00,00,02,64,00,00,02,6a,00,00
446 An DATA 02,70,00,00,02,78,00,00,02,8a
447 E3 DATA 00,00,02,90,00,00,02,96,00,00
448 hh DATA 02,9c,00,00,02,a8,00,00,02,d2
449 gX DATA 00,00,02,d8,00,00,02,e4,00,00
450 8r DATA 02,f0,00,00,02,fc,00,00,03,0a
451 Qc DATA 00,00,03,10,00,00,03,18,00,00
452 xE DATA 03,1e,00,00,03,36,00,00,03,44
453 8t DATA 00,00,03,4a,00,00,03,52,00,00
454 KM DATA 03,58,00,00,03,6a,00,00,03,74
455 Hr DATA 00,00,03,84,00,00,03,8a,00,00
456 Bv DATA 03,96,00,00,03,9c,00,00,03,ac
457 YQ DATA 00,00,03,b8,00,00,03,c2,00,00
458 re DATA 03,d2,00,00,03,e6,00,00,03,fc
459 so DATA 00,00,04,0c,00,00,04,12,00,00
460 e8 DATA 04,24,00,00,04,32,00,00,04,38
461 31 DATA 00,00,04,00,00,04,40,00,00,04
462 jr DATA 04,4e,00,00,04,5c,00,00,04,62
463 ec DATA 00,00,04,6a,00,00,04,70,00,00
464 uC DATA 04,7e,00,00,04,84,00,00,04,8c
465 S1 DATA 00,00,04,92,00,00,04,a0,00,00
466 m5 DATA 04,ae,00,00,04,bc,00,00,04,c2
467 De DATA 00,00,04,ca,00,00,04,d0,00,00
468 BN DATA 04,dc,00,00,05,10,00,00,05,18
469 9S DATA 00,00,05,28,00,00,05,3c,00,00
470 1S DATA 05,50,00,00,05,5e,00,00,05,6c
471 ZB DATA 00,00,05,7a,00,00,05,88,00,00
472 10 DATA 05,a2,00,00,05,a8,00,00,05,ba
473 GF DATA 00,00,05,c0,00,00,05,ee,00,00
474 cU DATA 05,ea,00,00,05,fa,00,00,05,00
475 JG DATA 00,00,06,08,00,00,06,0e,00,00
476 MS DATA 06,1e,00,00,06,32,00,00,06,3e
477 Ff DATA 00,00,06,4a,00,00,06,4e,00,00
478 So DATA 06,54,00,00,06,5a,00,00,06,70
479 KQ DATA 00,00,06,7a,00,00,06,a0,00,00


```

480 ZQ DATA 06,aa,00,00,06,b0,00,00,06,b4
481 Qk DATA 00,00,06,c0,00,00,06,cc,00,00
482 oO DATA 06,f4,00,00,07,1e,00,00,07,26
483 2j DATA 00,00,07,2c,00,00,07,32,00,00
484 oB DATA 07,56,00,00,07,64,00,00,07,6c
485 AH DATA 00,00,07,8a,00,00,07,c6,00,00
486 od DATA 07,cc,00,00,07,e8,00,00,07,ee
487 t3 DATA 00,00,07,fa,00,00,08,00,00,00
488 T9 DATA 08,40,00,00,08,4a,00,00,08,52
489 p7 DATA 00,00,08,5e,00,00,08,6a,00,00
490 NJ DATA 08,74,00,00,08,7a,00,00,08,8c
491 ga DATA 00,00,08,92,00,00,08,aa,00,00
492 ae DATA 08,ba,00,00,08,c0,00,00,08,c8
493 fO DATA 00,00,08,ce,00,00,08,d4,00,00
494 VB DATA 08,de,00,00,08,ee,00,00,08,f2
495 NY DATA 00,00,08,fc,00,00,09,02,00,00
496 TX DATA 09,08,00,00,09,14,00,00,09,1a
497 vR DATA 00,00,09,20,00,00,09,2a,00,00
498 zR DATA 09,6e,00,00,09,72,00,00,09,7c
499 Fq DATA 00,00,09,82,00,00,09,88,00,00
500 Nv DATA 09,8e,00,00,09,92,00,00,09,9c
501 Eq DATA 00,00,09,a2,00,00,09,a8,00,00
502 gD DATA 09,ae,00,00,09,b2,00,00,09,bc
503 CN DATA 00,00,09,c2,00,00,09,ca,00,00
504 9q DATA 09,d6,00,00,09,da,00,00,09,e4
505 LY DATA 00,00,09,ea,00,00,09,fe,00,00
506 5G DATA 0a,0a,00,00,0a,0e,00,00,0a,18
507 F1 DATA 00,00,0a,1e,00,00,0a,24,00,00

```

```

508 5X DATA 0a,4c,00,00,0a,56,00,00,0a,5a
509 N4 DATA 00,00,0a,60,00,00,0a,66,00,00
510 WL DATA 0a,70,00,00,0a,74,00,00,0a,7a
511 tY DATA 00,00,0a,82,00,00,0a,88,00,00
512 zx DATA 0a,98,00,00,0a,a2,00,00,0a,a6
513 pO DATA 00,00,0a,b0,00,00,0a,b6,00,00
514 gb DATA 0a,bc,00,00,0a,c2,00,00,0a,c6
515 9c DATA 00,00,0a,d0,00,00,0a,d6,00,00
516 L5 DATA 0a,de,00,00,0a,e8,00,00,0a,ee
517 dp DATA 00,00,0a,f4,00,00,0a,fe,00,00
518 FT DATA 0b,04,00,00,0b,0a,00,00,0b,10
519 GH DATA 00,00,0b,16,00,00,0b,36,00,00
520 yW DATA 0b,3a,00,00,0b,44,00,00,0b,4a
521 vd DATA 00,00,0b,62,00,00,0b,68,00,00
522 VW DATA 0b,72,00,00,0b,80,00,00,0b,86
523 fN DATA 00,00,0b,9e,00,00,0b,ba,00,00
524 K6 DATA 0b,c4,00,00,0b,d0,00,00,0b,de
525 D9 DATA 00,00,0b,ec,00,00,0b,fc,00,00
526 Vw DATA 0c,02,00,00,0c,0a,00,00,0c,12
527 dD DATA 00,00,0c,1a,00,00,0c,22,00,00
528 oH DATA 0c,34,00,00,0c,50,00,00,0c,5a
529 yJ DATA 00,00,0d,00,00,0d,0a,00,00,0d,10
530 sF DATA 0d,1e,00,00,0d,24,00,00,0d,3e
531 fV DATA 00,00,00,00,00,00,03,f2
(C) 1988 M&T

```

Listing 1. (Schluß)

```

59 mX cmp.l MemLaenge,d6
60 HQ bne trackfehler
61 wB jsr swapbuffer
;Fastmem<->Chipmem
62 1v move.l #0,Blk_Nr
63 Az jsr losablock
;Block 0-879 laden
64 rc cmp.l MemLaenge,d6
65 MV bne trackfehler
66 ezO ;***** Blockchangerou
tine *****
67 OL5 move.l #ctxt,text
68 L1 move.l #endtxtl-ctxt,textle
n
;Changing ausgeben
69 kt jsr printtxt
70 gnO taskoff:
71 zC5 move.l ExecBase,a6
;Taskswitching
72 vQ jsr Forbit(a6)
;verbieten
73 O1 move.l #7056,d0
74 CF clr.l d1
75 3W jsr AllocMem(a6)
76 F1 tst.l d0
77 y1 beq memfehler
78 tI move.l d0,Listadr
79 Fy move.l Fastbuffer,a0
80 1A move.l 316(a0),d0
81 6B move d0,BitmapBlock
;Adresse des Bitmapbloc
kes
82 Sw move.l #879,316(a0)
83 nS jsr AdrRech
;feststellen
84 Xn move.l a0,BMADR
85 j1 move.l #879,d1
86 ZO moveq #1,d2
87 WX jsr BlockMove
;Bitmap nach Block 881
88 yb move.l ABADR,BMADR
89 J4 move d1,BitmapBlock
90 QOO DiskSort:
91 fS5 move.l #881,d7
92 JR move.l d7,d4
93 5o move.l #880,d0
;RootBlock sortieren
94 3S0 DSSLoop:
95 Xa5 clr.l d6
96 oG moveq #2,d5
97 3V move d7,Firstsub
98 1W jsr SeHashB
99 gOO keinUDDS:
100 CH5 move.l d4,d0
101 fB addq.l #1,d4
102 IO cmp.l #1760,d4
103 gh bne.s weiterUDDS
104 wO moveq #2,d4
105 N1O weiterUDDS:
106 aE5 jsr SBitMap
107 I8 bne.s FertigDiskS
108 Pz cmp.l #879,d4
109 fY beq.s FertigDiskS
110 IU jsr AdrRech
111 aV move.l (a0),d2
112 pW cmp.l #8,d2
113 Sh beq.s keinUDDS
114 ru sub.l 508(a0),d2
115 Zg beq.s DSSLoop
;UserDir sortieren
116 6I bra.s keinUDDS
117 vqO FertigDiskS:
118 ep5 jsr ChkSumm
119 HBO exitchange:
120 Ew5 move.l ExecBase,a6
121 31 move.l Listadr,a1
122 nV move.l #7056,d0
123 Qq jsr FreeMem(a6)
124 YI bra taskan
125 cDO ;***** BlockMove: in d0 alte B1
ocknummer *****

```

Listing 2. Der Quelltext von FastLoadCopy

Programmname: FastLoadCopy
Computer: A500, A1000, A2000 mit Kickstart 1.2
Sprache: Assembler
Assembler: DevPac

Programmname: FastLoadCopy

```

1 XS0 ;*****
*****
2 Kp ;* FastLoadCopy b
y Paul Sponagl *
3 s3 ;* (C) Markt &
Technik *
4 No ;* in July
1988 *
5 59 ;* TEL: 083
68/1581 *
6 cX ;*****
*****
7 DS ;***** Alle Makros d
effinieren *****
8 11 ExecBase equ 4
9 jo FindTask equ -294
10 Fy AddPort equ -354
11 Vg RemPort equ -360
12 fy OpenLib equ -408
;Exec-Kommandos
13 b6 CloseLib equ -414
14 Y3 Forbit equ -132
15 Iy Permit equ -138
16 nU AllocMem equ -198
17 bm FreeMem equ -210
18 tD ChipMem equ 3
; " (Speicher)
19 X8 FastMem equ 5
20 Qf OpenDev equ -444
21 ST CloseDev equ -450
; " (Trackdisk)
22 Wy DoIo equ -456
23 Xo td_read equ 2
24 xL td_format equ 11
;Trackdisk-Kommandos
25 Ae OutPut equ -60
26 HH Write equ -48
;DOS-Kommandos

```

```

27 KT Exit equ -144
28 1k ;***** Start
*****
29 9n start:
30 Tw5 move.l a7,SPointer
31 jP clr.b Errorstate
32 mv move.l #clstxt,text
33 4r move.l #memtxt-clstxt,txtl
en
;Anfangstext
34 BK jsr printtxt
35 dQ move.l #quelltxt,text
36 s8 move.l #zieltxt-quelltxt,te
xtlen
;Quelldiskette!
37 EN jsr printtxt
38 Ep jsr wait
39 M9O ;***** Ganze Diskette in
den Speicher laden *****
40 lg5 move.l #ChipMem,Bedingung
41 gr move.l #450560,MemLaenge
;ChipMem fuer 880 Bloec
ke
42 wd jsr getmem
;reservieren
43 k1 tst.l Buffer
44 RU beq memfehler
45 gV move.l Buffer,Chipbuffer
46 wa move.l #FastMem,Bedingung
47 SX jsr getmem
;FastMem ebenfalls
48 pq tst.l Buffer
49 WZ beq memfehler
50 ZJ move.l Buffer,Fastbuffer
51 Tx move.l #1txt,text
52 nH move.l #ctxt-ltxt,txtlen
; "loading"ausgeben
53 Ud jsr printtxt
54 HJ move.l MemLaenge,bufflen
55 do move.l Chipbuffer,diskbuff
56 zr move #td_read,Kommando
;Block 880-1759 laden
57 Xu move.l #880,Blk_Nr
58 zH jsr losablock

```

```

126 rI ; in d1 neue Bl
; ocknummer *
127 Jm ; in d2 Blockar
t *
128 6r ;*****
*****
BlockMove:
129 On move.m 1 d0-d6/a0-a6,-(a7)
130 t15 jsr AdrRech
131 dp move.l a0,ABADR
132 ko move.l d0,ABNR
133 Or ;Adressen fuer Ver-
; schiebungsroutinen
134 rf move.l d1,NBNR
135 WY move.l d1,d0
136 iu jsr AdrRech
137 FW move.l a0,NBADR
138 JE jsr WhatMove
; 1.Block(d0)
139 Nv move.l ABNR,d0
140 Aw move.l NBNR,ABNR
141 RO move.l d0,NBNR
142 c0 move.l ABADR,d0
; Adressen und
143 ee move.l NBADR,ABADR
; Nummern austauschen
144 Ys move.l d0,NBADR
145 Oy clr offset
146 KF move.l #0,a0
147 n2 jsr Listeintrag
148 W4 move.l ABNR,d0
149 Hv jsr SBitMap
150 8U bne.s blockleer
; Tauschblock leer?
151 KF move.l ABADR,a0
152 FA move.l (a0),d2
153 Wr cmp.b #8,d2
154 fA beq.s weiterBM
155 WZ sub.l 508(a0),d2
156 lD0 weiterBM:
157 kc5 jsr WhatMove
; 2.Block(d1)
158 yY0 blockleer:
159 ma5 jsr Blockeintrag
160 tY jsr SwapBit
; Blocks u. BBits austau
schen
161 sm jsr SwapBlock
162 h9 movem.l (a7)+,d0-d6/a0-a6
163 L1 rts
164 O10 WhatMove:
165 nL5 move.l ABNR,d6
166 Um cmp.b #5,d2
167 If beq.s FHMov
168 l6 cmp.b #8,d2
169 Yv beq DBMov
170 u0 tst.b d2
; Blockroutine auswaehle
n
171 qW beq UDMov
172 Oa cmp.b #19,d2
173 Y7 beq FLMov
174 IW cmp.b #1,d2
175 hX beq.s weiterWM
176 Vf move.b #8,Errorstate
177 LB jsr changeerror
178 Ov0 weiterWM:
179 bH5 rts
180 t90 ;***** Blockerschiebungsrout
inen *****
181 pa ;* benoetigen die Adressen NBNR,A
BNR,ABADR *
182 mh ;*****
*****
183 mQ ;***** File-Header versch
. *****
184 xm FHMov:
185 qz5 movem.l d0-d7/a0-a6,-(a7)
186 6t jsr HKey
; Neuer Header-Key
187 l4 clr.l d5
188 ym move.l 500(a0),d0
189 Ez jsr SeHashB

190 2x move.l a0,a1
191 xf move.l 16(a0),d0
192 WE0 FHLoop1:
193 dp5 jsr AdrRech
194 6z cmp.l 4(a0),d6
195 NF beq.s noerrorFH2
196 QV move.b #3,Errorstate
197 O6 jsr changeerror
; In alle Datenbl.
198 Tq0 noerrorFH2:
199 QS5 move #4,offset
200 et jsr Listeintrag
201 7p move.l 16(a0),d0
202 Hk tst.l d0
203 Gp bne.s FHLoop1
204 LH move.l a1,a0
205 mV0 FHLoop2:
206 eC5 move.l 504(a0),d0
207 Mp tst.l d0
208 71 beq.s fertigFH
209 t5 jsr AdrRech
210 d2 cmp.l 500(a0),d6
211 P1 beq.s noerrorFH
; In alle FileList
212 lr move.b #4,Errorstate
213 v1 jsr changeerror
214 zW0 noerrorFH:
215 x75 move #500,offset
216 u9 jsr Listeintrag
217 Qz bra.s FHLoop2
218 xZ0 fertigFH:
219 hA5 movem.l (a7)+,d0-d7/a0-a6
220 Gw rts
221 iV0 ;***** File-List versch.
*****
222 le FLMov:
223 Sb5 movem.l d0-d7/a0-a6,-(a7)
224 iV jsr HKey
; Neuer Header-Key
225 ZN move.l 500(a0),d0
226 Rw0 FLLoop:
227 BN5 jsr AdrRech
228 OY move.l 504(a0),d0
229 iB tst.l d0
230 JU bne.s noerrorFL
231 9C move.b #5,Errorstate
232 E4 jsr changeerror
233 cD0 noerrorFL:
234 GH5 cmp.l d0,d6
; Neue Blocknummer in
235 DB bne.s FLLoop
; Fileheader(Hash)
236 Qe move #504,offset
237 FU jsr Listeintrag
238 OT movem.l (a7)+,d0-d7/a0-a6
239 ZF rts
240 vC0 ;***** User-Dir versch.
*****
241 AA UDMov:
242 lu5 movem.l d0-d7/a0-a6,-(a7)
243 l0 jsr HKey
; Neuer Header-Key
244 sg move.l 500(a0),d0
245 x0 clr.l d5
246 OB jsr SeHashB
; Neue Hashtab.
247 Ab moveq #1,d5
248 lN move.l d6,d0
; Alle Unterdirs
249 Cx jsr SeHashB
250 Cf movem.l (a7)+,d0-d7/a0-a6
251 lR rts
252 ny0 ;***** Data-Block vers
ch. *****
253 iP DBMov:
254 x65 movem.l d0-d7/a0-a6,-(a7)
255 Ov move.l ABADR,a0
256 n5 move.l 4(a0),d0
257 SW move.l 8(a0),d1
; Erster Datenblock?
258 gY cmp.w #1,d1
259 np beq.s erstblock
260 Gq0 DBLoop1:

261 jv5 jsr AdrRech
262 d0 adda.l #24,a0
; Blocknummer eintragen
263 sH moveq #71,d1
264 Hv0 SHLoop:
265 Ct5 cmp.l (a0)+,d6
266 lY dbeq d1,SHLoop
267 XM beq.s gefunden
; Neue Hashtabelle
268 5y move.l SBLOCK,a0
269 fD move.l 504(a0),d0
270 KF move.l a0,a2
271 g7 bra.s DBLoop1
272 Vh0 gefunden:
273 d65 move #4,offset
274 q5 jsr Listeintrag
275 Nt tst.l (a0)
276 lY beq.s Vblock
; Noch NextData eintrage
n
277 GB move.l (a0),d0
278 Qu bra.s weiterDB
279 cw0 Vblock:
280 O75 move.l 24(a2),d0
281 QT0 weiterDB:
282 4G5 jsr AdrRech
283 k3 cmp.l 16(a0),d6
284 mo beq.s noerrorDB2
; Vorgangerblock
285 6E move.b #6,Errorstate
286 6w jsr changeerror
287 iX0 noerrorDB2:
288 On5 move #16,offset
289 5K jsr Listeintrag
290 s0 bra.s fertigDB
291 310 erstblock:
292 EQ5 jsr AdrRech
293 cT cmp.l 308(a0),d6
294 aU beq.s noerrorDB1
295 G0 move.b #6,Errorstate
296 G6 jsr changeerror
297 Mf0 noerrorDB1:
298 q85 move #308,offset
; In alle Filelist u.
299 FU jsr Listeintrag
300 xY0 DBLoop2:
301 2L5 cmp.l 16(a0),d6
302 cy beq.s noerrorDB
303 z2 move.b #1,Errorstate
304 OE jsr changeerror
305 qF0 noerrorDB:
306 DY5 move #16,offset
; Fileheader eintragen
307 Ne jsr Listeintrag
308 lq move.l 504(a0),d0
309 OT tst.l d0
310 bA beq.s fertigDB
311 XJ jsr AdrRech
312 Lm bra.s DBLoop2
313 Ou0 fertigDB:
314 Eh5 movem.l (a7)+,d0-d7/a0-a6
315 nT rts
316 6z0 HKey:
317 Ov5 move.l ABADR,a0
318 6z cmp.l 4(a0),d6
319 kn beq.s weiterHK
; Neuen HeaderKey
320 kt move.b #7,Errorstate
321 fV jsr changeerror
322 r70 weiterHK:
323 Q55 move #4,offset
324 et jsr Listeintrag
325 xd rts
326 gn0 ;***** SBitMap: Blocknummer i
n d0 *****
SBitMap:
327 Ni movem.l d0,-(a7)
328 Kz5 move.l BMADR,d5
329 OY moveq #2,d1
330 a2
331 uH0 SBM1:
332 dC5 addq.l #4,d5
333 qB add.w #32,d1
334 Tv cmp.w d1,d0

```



```

;Bit in Bitmap testen
335 rm      bpl.s  SBM1
336 K7      sub.w  d0,d1
337 J2      exg.l  d5,a0
338 FA      move.l (a0),d5
339 D5      sub.w  #32,d1
340 r5      neg.w  d1
341 wk      btst  d1,d5
342 tp      movem.l (a7)+,d0
343 Fv      rts
344 pk0     BMADR ds.l 1
345 66      ;***** AdrRech: Blocknummer i
n d0 *****
346 a8      AdrRech:
347 d15     movem.l d0,-(a7)
348 9o      cmp.w  #880,d0
349 hf      bpl.s  groesserAR
350 tg      move.l Chipbuffer,a0
351 Mn      bra.s  weiterAR
352 R10     groesserAR:
353 f05     move.l Fastbuffer,a0
354 45     sub.w  #880,d0
;Aus Blocknummer Speic
her-
355 V10     weiterAR:
356 Gu5     mulu  #512,d0
;adresse berechnen
357 W1      adda.l d0,a0
358 xP      move.l a0,SBLOCK
359 A6      movem.l (a7)+,d0
360 WC      rts
361 PRO     SBLOCK ds.l 1
362 XC      ;**** SeHashB: Bedingung in d5 B
lkn. in d0 ****
363 fe      SeHashB:
364 895     movem.l d0-d4/a0-a1,-(a7)
365 Pb      jsr  AdrRech
366 qk      adda.l #24,a0
367 Yx      moveq  #71,d3
368 KW0     SHBLoop:
369 Zf5     move.l (a0)+,d0
370 zS      tst.l  d0
371 VF      beq.s  weiterSHB1
372 9u      jsr  UDeintrag
373 Y9      cmp.l  #2,d5
374 1r      bne.s  keinsort
;Diese Routine durch-
375 ex      move.l ABNR,d0
;sucht die Hashtabelle
376 Ta      bra.s  schonsort
;und fuegt neue Werte e
in
377 oA0     keinsort:
378 fH5     cmp.l  d0,d6
379 gb      beq.s  gefundenSHB1
380 B10     schonsort:
381 725     move.l a0,a1
382 iC0     SHBLoop2:
383 ht5     jsr  AdrRech
384 Nv      move.l 496(a0),d0
385 Eh      tst.l  d0
386 kU      beq.s  weiterSHB2
387 8b      jsr  UDeintrag
;Such nach HashChain
388 n0      cmp.l  #2,d5
389 dK      bne.s  keinsort2
390 Qy      move.l ABNR,d0
391 MU      bra.s  SHBLoop2
392 V90     keinsort2:
393 uW5     cmp.l  d0,d6
394 vq      beq.s  gefundenSHB2
395 QY      bra.s  SHBLoop2
396 se0     weiterSHB2:
397 S05     move.l a1,a0
398 oe0     weiterSHB1:
399 x15     dbra  d3,SHBLoop
400 Tw      tst.l  d5
401 oz      bne.s  fertigSHB
402 qS0     obenSHB:
403 AK5     move.b #8,Errorstate
404 Oq      jsr  changeerror
405 Yo0     gefundenSHB1:
406 NN5     suba.l #500,a0
407 du0     gefundenSHB2:
408 b45     tst.l  d5
409 mh      bne.s  obenSHB
410 5k      move  #496,offset
411 3I      jsr  Listeintrag
412 Sp0     fertigSHB:
413 Cn5     cmp.l  #2,d5
414 7q      bne.s  fertigSHB2
415 U1      jsr  FileSort
416 GT0     fertigSHB2:
417 e45     movem.l (a7)+,d0-d4/a0-a1
418 S8      rts
419 hG0     UDeintrag:
420 nG5     tst.l  d5
421 zT      beq.s  weiterUDE
422 Lw      cmp.l  #2,d5
423 ap      beq.s  DirSort
424 ef      movem.l d0/a0,-(a7)
425 NZ      jsr  AdrRech
426 oc      move.l 500(a0),d0
427 aZ      cmp.l  d0,d6
;Routine fuer Userdir
428 SI      beq.s  noerrorUDE
429 6A      move.b #2,Errorstate
430 QG      jsr  changeerror
431 i20     noerrorUDE:
432 Sc5     move  #500,offset
433 Pe      jsr  Listeintrag
434 oI      movem.l (a7)+,d0/a0
435 qn0     weiterUDE:
436 kQ5     rts
437 Vz0     DirSort:
438 5G5     movem.l d0-d2/a0,-(a7)
439 bn      jsr  AdrRech
440 to      move.l (a0),d2
441 AV      cmp.b  #8,d2
442 Bp      beq.s  weiterDiSo
443 AD      sub.l  508(a0),d2
444 KJ0     weiterDiSo:
445 O85     move.l d7,d1
446 MY      jsr  BlockMove
;Alle Subeintraege!
447 F1      addq.l #1,d7
448 Gm      addq.l #1,d6
449 WF      jsr  Newnum
450 2g      cmp.l  #5,d2
451 cP      bne.s  keinFH
452 2k      move.l ABNR,Bbuff
453 bD0     FLSORTLoop:
454 S05     move.l ABNR,d0
455 r3      jsr  AdrRech
456 gE      move.l 504(a0),d0
457 Or      tst.l  d0
458 4u      beq.s  keinFL
459 Id      move.l #19,d2
460 34      move.l d7,d1
;Alle FileList nach
461 nW      jsr  BlockMove
;FileHeader eintr.
462 U0      addq.l #1,d7
463 V1      addq.l #1,d6
464 1U      jsr  Newnum
465 vY      bra.s  FLSORTLoop
466 NO0     keinFL:
467 1L5     cmp.b  #19,d2
468 oh      beq.s  mehrdata
469 Uq      move.l 8(a0),d0
470 2c      cmp.l  #1,d0
471 WM      bne.s  mehrdata
472 UC      move.l 16(a0),d0
473 H1      move.l d7,d1
;1-Block Data versch.
474 MS      move.l #8,d2
475 iz      jsr  BlockMove
476 iE      addq.l #1,d7
477 JF      addq.l #1,d6
478 zi      jsr  Newnum
479 ke0     mehrdata:
480 ZZ5     move.l Bbuff,ABNR
481 URO     keinFH:
482 gY5     movem.l (a7)+,d0-d2/a0
483 VB      rts
484 uh0     FileSort:
485 XU5     move  Firstsub,d0
486 cv      ext.l  d0
487 8n0     FSLoopa:
488 tM5     tst.l  d6
489 OP      beq.s  kleinerFHFS
490 Qc      jsr  AdrRech
491 id      move.l (a0),d1
492 xe      cmp.l  #8,d1
493 HN      beq.s  keinFHFS
494 z2      sub.l  508(a0),d1
495 cW      cmp.l  #5,d1
;Alle Datenbloecke
496 jn      bne.s  keinFHFS
;des Parentdir
497 tc      move.l 8(a0),d1
;suchen...
498 U4      cmp.l  #1,d1
499 NT      beq.s  keinFHFS
500 Xu      jsr  Filesortb
501 wK0     keinFHFS:
502 8e5     addq.l #1,d0
503 a0      cmp.l  #1759,d0
504 G0      bls.s  kleinerFHFS
505 Pr      moveq  #2,d0
506 L60     kleinerFHFS:
507 YN5     dbra  d6,FSLoopa
508 ua      rts
509 JW0     Filesortb:
510 FJ5     movem.l d0-d2,-(a7)
511 ku0     FSBLoop:
512 8q5     move.l 16(a0),d0
513 yH      beq.s  fertigFSB
514 7F      move.l d7,d1
515 17      move.l #8,d2
516 eC      jsr  BlockMove
;...und sortieren!
517 Nt      addq.l #1,d7
518 dM      jsr  Newnum
519 V3      move.l ABNR,d0
520 u6      jsr  AdrRech
521 3r      bra.s  FSBLoop
522 Je0     fertigFSB:
523 Lq5     movem.l (a7)+,d0-d2
524 Aq      rts
525 uB0     Newnum:
526 x15     cmp.l  #1759,d7
527 7V      bls.s  kleinerNN
528 zH      moveq  #2,d7
;Bereichstest
529 O00     kleinerNN:
530 Gw5     rts
531 2Q0     Firstsub ds.w 1
532 uG      Bbuff ds.l 1
533 F6      ;***** ChkSumm: Alle Checksummen
berechnen *****
534 cN      ChkSumm:
535 oF5     moveq  #1,d0
536 gF0     CSLoop:
537 3k5     addq.w #1,d0
538 Yg      cmp.w  BitmapBlock,d0
539 md      beq.s  CSBitMap
540 qo      cmp.w  #1760,d0
541 Kw      beq.s  FertigCS
;Berechnen aller Check-
542 Fp      jsr  SBitMap
;summen ausser Bootblo
cke
543 Hg      bne.s  CSLoop
544 IU      jsr  AdrRech
545 nq      clr.l  d4
546 5h      moveq  #127,d3
547 mV0     CHKS:
548 Uo5     add.l  (a0)+,d4
549 hl      dbra  d3,CHKS
550 HJ      sub.l  -492(a0),d4
551 C8      neg.l  d4
552 O0      move.l d4,-492(a0)
553 Nm      bra.s  CSLoop
554 u80     FertigCS:
555 fL5     rts
556 X10     CSBitMap:

```

Listing 2. (Fortsetzung)

```

557 Vh5      jsr   AdrRech
558 Fq       moveq #126,d3
559 uA       adda.l #4,a0
560 25       clr.l d4
561 d50     CHKB:
562 125      add.l (a0)+,d4
563 vz       dbra d3,CHKB
564 PL       neg.l d4
565 81       move.l d4,-512(a0)
566 az       bra.s CSLoop
567 zN0      ABNR ds.l 1
568 Q1       NBNR ds.l 1
569 tc       ABADR ds.l 1
570 KG       NBADR ds.l 1
571 Ph       BitmapBlock ds.w 1
572 O1       offset ds.w 1
573 QQ       Listadr ds.l 1
574 Om       Listzaehl ds.w 1
575 v8       ;***** SwapBlock: B.adr. in a
           0 u. a1 *****
576 jt       SwapBlock:
577 NL5      movem.l d0-d1/a0-a1,-(a7)
578 D8       move.l ABADR,a0
579 4C       move.l NBADR,a1
580 dF       moveq #127,d0
581 6e0     SBLoop:
582 B65      move.l (a0),d1
583 VY       move.l (a1),(a0)+
584 XS       move.l d1,(a1)+
           ;Bloecke austauschen
585 L1       dbra d0,SBLoop
586 8V       movem.l (a7)+,d0-d1/a0-a1
587 Br       rts
588 i10     ;***** SwapBit: B.nr. ABNR
           u.NBNR *****
589 ic       SwapBit:
590 JR5      movem.l d0-d6/a0-a6,-(a7)
591 s0       move.l ABNR,d0
592 D0       jsr   SBitMap
593 71       move.l a0,a6
594 md       move.l d1,d4
           ;Belegungsbits vertausc
           hen
595 10       move.l d5,d6
596 OJ       move.l NBNR,d0
597 I5       jsr   SBitMap
598 NL       cmp.l d5,d6
599 QG       beq.s gleichSB
600 6J       btst d4,d6
601 hV       beq.s belegt1
602 t3       bset d1,d5
603 Vh0     weiter1:
604 kY5      beq.s belegt2
605 EC       bset d4,d6
606 Pd       bra.s weiter2
607 8h0     belegt1:
608 VU5      bclr d1,d5
609 Sg       bra.s weiter1
610 Eo0     belegt2:
611 qd5      bclr d4,d6
612 hu0     weiter2:
613 TD5      move.l d6,(a6)
614 uc       move.l d5,(a0)
615 uk0     endeSB:
616 1T5      movem.l (a7)+,d0-d6/a0-a6
617 fL       rts
618 kc0     gleichSB:
619 Pc5      btst d4,d6
620 eM       beq.s belegt1g
621 CM       bset d1,d6
622 OJ0     weiter1g:
623 nR5      beq.s belegt2g
624 XV       bset d4,d6
625 ZJ       bra.s weiter2g
           ;Worte aus Bitmap
626 iJ0     belegt1g:
627 x35      bclr d1,d6
           ;gleich!
628 FX       bra.s weiter1g
629 7q0     belegt2g:
630 9w5      bclr d4,d6
631 aw0     weiter2g:
632 mW5      move.l d6,(a6)

```

```

633 b1       bra.s endeSB
634 jv0     Listeintrag:
635 v25      movem.l d0/a0-a1,-(a7)
636 Bh       move offset,d0
637 3M       ext.l d0
638 3I       adda.l d0,a0
639 h5       move Listzaehl,d0
640 6P       ext.l d0
641 rZ       move.l Listadr,a1
           ;Alle Bloecke in Liste
642 7M       adda.l d0,a1
643 k4       move.l a0,(a1)
644 fE       addq.l #4,d0
645 XE       move d0,Listzaehl
646 sp       movem.l (a7)+,d0/a0-a1
647 9p       rts
648 170     Blockeintrag:
649 rF5      move Listzaehl,d3
650 Fn       lsr #2,d3
651 OJ       subq #1,d3
652 ca       move.l Listadr,a0
653 fD       move.l ABNR,d1
654 sC0     BELoop:
655 V15      tst.l (a0)
656 1G       beq.s weiterBE
657 y1       move.l (a0)+,a1
658 Ge       move.l d1,(a1)
659 NZ       dbra d3,BELoop
660 FP0     loeschen:
661 Cj5      clr Listzaehl
662 cM       move.l Listadr,a0
           ;Alle Blockaenderungen
663 OQ       move #1763,d3
           ;aus Liste eintragen
664 2M0     clrloop:
665 EW5      clr.l (a0)+
666 rJ       dbra d3,clrloop
667 T9       rts
668 lp0     weiterBE:
669 Kw5      adda.l #4,a0
670 Cv       move.l NBNR,d1
671 Er       dbra d3,BELoop
672 7n       bra.s loeschen
673 cg0     changeerror:
674 js5      movem.l d0-d7/a0-a6,-(a7)
675 1Z       move.l ABNR,d0
676 Cc       lea bnummer,a0
677 uh       moveq #2,d3
           ;Blocknummer ausgeben
678 wE       jsr numtxt
679 d1       lea enummer,a0
           ;Fehlernummer ausgeben
680 31       move.b Errorstate,d0
681 pQ       ext.w d0
682 OJ       clr.l d3
683 1J       jsr numtxt
684 K2       move.l ExecBase,a6
685 Qw       jsr Permit(a6)
686 pv       move.l #cerrtxt,text
687 kE       move.l #text-cerrtxt,textle
           n
           ;schreiben!
688 js       jsr printtxt
689 Cr       move.b Errorstate,d0
690 6Q       cmp.b #7,d0
691 ah       bls.s geht
692 u5       jsr ChkSumm
693 RI       jsr swapbuffer
694 Y9       move.l SPointer,a7
           ;Hard Error (F.Nr. 8)
695 no       bra exitchange
696 H30     geht:
697 T95      clr.b Errorstate
698 Qf       movem.l (a7)+,d0-d7/a0-a6
699 zt       rts
700 bs0     Errorstate ds.b 1
701 zC       SPointer ds.l 1
702 15      ;***** numtxt: Nummer i
           n d0 *****
703 wq       numtxt:
704 Rd5      movem.l d0-d3/a0,-(a7)
705 Nf       adda.l d3,a0
706 2F       adda.l #1,a0
707 690     nummloop:

```

```

708 wf5      clr d2
709 ta       move d0,d2
710 ze       and #15,d2
711 Qa       add #48,d2
           ;Nummer in Register
712 Hd       cmp #9',d2
           ;d0 ab Adresse a0
713 P9       bls ziffer
           ;mit Laenge-1 in d3
714 hg       add #7,d2
715 SD0     ziffer:
716 kB5      move.b d2,-(a0)
717 U4       lsr #4,d0
718 72       dbra d3,nummloop
719 aT       movem.l (a7)+,d0-d3/a0
720 KO       rts
721 Jg0     ;***** Fehlerausga
           be *****
722 pv       memfehler:
723 rW5      move.l #memtxt,text
724 fx       move.l #quelltxt-memtxt,tex
           tlen
           ;Fehlermeldung
725 KT       jsr printtxt
726 d4       jmp schluss
727 xQ0     trackfehler:
728 DC5      move.l #losatxt,text
           ;Fehlermeldung
729 u4       move.l #stxt-losatxt,textle
           n
730 PY       jsr printtxt
731 19       jmp schluss
732 e50     ;***** Programme
           nde *****
733 mM       ende:
734 jP5      move.l #endtxt1,text
735 sD       move.l #endtxt2-endtxt1,tex
           tlen
           ;1.Meldung erstellen
736 Ve       jsr printtxt
737 vc       clr.b d0
738 9B       jsr tastloop
739 oS       tst.b d0
           ;L o. R Taste?
740 HT       beq schluss
741 wY       move.l #endtxt2,text
742 6s       move.l #cerrtxt-endtxt2,tex
           tlen
           ;R. Taste!
743 cl       jsr printtxt
744 Ke0     tastloop2:
745 B55      btst #10,$dff016
746 Iu       beq tastloop2
           ;R. Taste loslassen
747 5m       clr.b d0
748 JL       jsr tastloop
749 OG       tst.b d0
           ;L. Taste:Neue Disk
750 5q       beq weiter
751 NW       move.l #clstxt,text
752 11       move.l #memtxt-clstxt,txtl
           en
           ;R. Taste:Multicopy
753 mv       jsr printtxt
754 QH       jsr swapbuffer
755 JJ       bra savedisk
756 460     tastloop:
757 8T5      btst #6,$bfe001
758 gk       beq links
759 PJ       btst #10,$dff016
760 Qn       bne tastloop
           ;Tastenauswertung
761 UL       add.b #1,d0
762 Ou0     links:
763 1h5      rts
764 wW0     schluss:
765 jv5      tst.l Fastbuffer
766 CI       beq nextbuff
767 AH       move.l Fastbuffer,Buffer
768 8o       jsr nomem
           ;Speicher freigeben
769 kG0     nextbuff:
770 5v5      tst.l Chipbuffer
771 B5       beq aus
772 WH       move.l Chipbuffer,Buffer
773 Fm       jsr nomem
774 3o0     aus:

```



```

775 Dt5      rts
776 T00 weiter:
777 sR5      jsr  schluss
778 tR      jmp  start
779 de0 taskan:
780 sa5      move.l ExecBase,a6
781 yU      jsr  Permit(a6)
782 rx      tst.b Errorstate
783 Jj      bne  ende
784 Y50 ;***** Ganze Diskette
zurueckspeichern *****
savedisk:
785 Oc      move.l #zieltxt,text
786 ys5      move.l #maustxt-zieltxt,tex
787 Gq      tlen ;Zieldiskette!
788 LU      jsr  printtxt
789 Lw      jsr  wait
790 xY      move.l #stxt,text
791 3f      move.l #ltxt-stxt,textlen
; "saving" ausgeben
792 PY      jsr  printtxt
793 r4      move #td_format,Kommando
794 pJ      move.l #0,Blk_Nr
795 FU      jsr  losablock
; Block 0-879 speichern
796 6x      jsr  swapbuffer
797 Tq      move.l #880,Blk_Nr
798 6v      jsr  losablock
; Block 880-1759 speiche
rn
799 UC      jmp  ende
800 yF0 swapbuffer:
801 AX5      move.l #1,Chipbuffer,a1
802 ud      move.l #Fastbuffer,a0
803 y5      move.l #112640,d0
804 Zt0 transbuff:
805 ZY5      move.l (a0),d1
; Chip <=> Fast
806 69      move.l (a1),(a0)+
807 Ov      move.l d1,(a1)+
808 LO      subq.l #1,d0
809 Jc      bne.s transbuff
810 mS      rts
811 Wf0 Chipbuffer ds.l 1
812 GZ      Fastbuffer ds.l 1
813 r3 ;***** Warten auf 1
. Maustaste *****
814 ZY wait:
815 Tj5      move.l #maustxt,text
816 Fa      move.l #losatxt-maustxt,tex
tlen
817 ox      jsr  printtxt
818 WR0 waitloop:
819 9q5      btst #6,$bfe001
; Textausgabe
820 y1      bne.s waitloop
; und warten
821 xd      rts
822 AP0 ;***** Speicher: nomem + getme
m *****
823 qq ; Routine belegt Speicherbereich
*
824 3w ; * Laenge in MemLaenge, Speichera
rt in Bed. *
825 Y7 ; * Rueckgabewerte:
*
826 dJ ; * Adresse in Buffer
*
827 bM ;*****
*****
828 mQ getmem:
829 fN5      move.l ExecBase,a6
830 aT      move.l MemLaenge,d0
; Speicher beantragen
831 LJ      move.l #Bedingung,d1
832 GJ      jsr  AllocMem(a6)
833 YC      move.l d0,Buffer
834 9e0 fehler:
835 Br5      rts
836 oJ0 nomem:
837 nV5      move.l ExecBase,a6
838 2C      move.l MemLaenge,d0
839 WO      move.l Buffer,a1
; Speicher freigeben
840 zP      jsr  FreeMem(a6)
841 Hx      rts
842 Fx0 Bedingung ds.l 1
843 Bz      Buffer ds.l 1
844 bi      MemLaenge ds.l 1
845 Dq ;***** Blockroutine: losabloc
k *****
846 2m ; * Uebergabewerte:
*
847 4i ; * Startblocknummer in Blk_Nr
*
848 zm ; * Speicher fuer Block(s) in
diskbuff *
849 Pg ; * Speicherlaenge in bufflen
*
850 IV ; * Anweisung in Kommando
*
851 yX ; * Rueckgabewerte:
*
852 Uq ; * Uebertragene Bytes in d6
*
853 lm ;*****
*****
854 g6 losablock:
855 OM5      lea  diskio,a6
856 a4      move.l #27,d0
857 7M0 clearloop:
858 Mu5      move.l #0,(a6)+
; Struktur loeschen
859 SW      dbra d0,clearloop
860 AS      move.l ExecBase,a6
861 OQ      move.l Blk_Nr,d0
; Blk_Nr mal 512
862 LQ      mulu #512,d0
863 85      move.l d0,Blk_Nr
864 Iz      sub.l a1,a1
865 Dq      jsr  FindTask(a6)
866 Jq      lea  readreply,a1
; Eigenen Task
867 aX      move.l d0,16(a1)
; wieder aufwecken
868 Gy      jsr  AddPort(a6)
869 ca      lea  diskio,a1
870 64      move.l #0,d0
871 36      clr.l d1
872 xW      lea  trddevice,a0
; Trackdiskdevice oeffne
n
873 WC      jsr  OpenDev(a6)
874 7a      tst.l d0
875 1s      bne  error
876 Jh      lea  diskio,a1
877 B9      move.l #readreply,14(a1)
878 oY      move Kommando,28(a1)
; Struktur beschreiben
879 XF      move.l diskbuff,40(a1)
; und Block laden
880 EB      move.l bufflen,36(a1)
881 46      move.l Blk_Nr,44(a1)
882 WE      move.l ExecBase,a6
883 VN      jsr  DoIo(a6)
884 rp      lea  diskio,a1
885 Oh      move.l 32(a1),d6
886 oc      move #9,28(a1)
; Motor ausschalten
887 Gs      move.l #0,36(a1)
888 aS      jsr  DoIo(a6)
889 XL      lea  readreply,a1
890 uY      jsr  RemPort(a6)
; Port entfernen
891 yw      lea  diskio,a1
892 oY      jsr  CloseDev(a6)
; Trddevice schliessen
893 Cw0 error:
894 8o5      rts
895 J40 trddevice dc.b 'trackdisk.device'
,0
896 Fg      diskio ds.l 20
897 UJ      readreply ds.l 8
898 tF      bufflen ds.l 1
899 V6      diskbuff ds.l 1
900 CJ      Blk_Nr ds.l 1
901 f8 Kommando ds.w 1
902 rk ;***** Textausgabe printtx
t: *****
903 5u ; * Textanfang in text
*
904 QI ; * Textlaenge in textlen
*
905 rc ;*****
*****
906 gR printtxt:
907 QY5      move.l d0-d6/a0-a6,-(a7)
908 we      move.l ExecBase,a6
909 tE      lea  dosname,a1
910 4t      clr.l d0
; Dos-Lib oeffnen
911 A1      jsr  OpenLib(a6)
912 5Q      beq  falsch2
913 SI      move.l d0,dosbase
914 tF      move.l d0,a6
; Outputkanal(Fenster) h
olen
915 CY      jsr  OutPut(a6)
916 23      move.l d0,d1
917 Cy      move.l text,d2
918 Sm      move.l textlen,d3
; Text ausgeben
919 gG      jsr  Write(a6)
920 rK      tst.l d0
921 Xf      bpl  close
922 lg      jmp  Exit(a6)
923 CF0 close:
924 Cu5      move.l ExecBase,a6
925 Oa      move.l dosbase,a1
926 eo      jsr  CloseLib(a6)
; Dos-Lib schliessen
927 dV0 falsch2:
928 3V5      move.l (a7)+,d0-d6/a0-a6
929 hN      rts
930 zk0 dosname dc.b 'dos.library',0
931 rI      dosbase ds.l 1
932 i5      clstxt dc.b 12,10,10,9,9,155
933 Co9      dc.b '1;31;42m FastLo
adCopy by P.S. (C) M&T
in 1988 '
934 n9      dc.b 155,'0m',10,10,10
935 Vw0 memtxt dc.b 'Zu wenig Speicher
vorhanden',10
936 OW      quelltxt dc.b 'Bitte Quelldiskett
e in DF0:',10
937 Jm      zieltxt dc.b 10,'Bitte Zieldiske
tte in DF0:',10
938 9A      maustxt dc.b '... und linke Maus
taste druecken!',10
939 yn      losatxt dc.b 'Trackdisk.device -
Probleme',10
940 ol      stxt dc.b '... saving',10,10
941 wK      ltxt dc.b '... loading',10,10
942 Fa      ctxt dc.b '... changing',10,1
0
943 E4      endtxt1 dc.b 10,10,10,9,155
944 Jm9      dc.b '0;31;42m LMaustas
te = Quit ',155,'0m
'
945 MB      dc.b 155,'0;31;42m RMau
staste = Nochmal ',155,
'0m',10
946 qo0      dc.b 10,9,155
947 R59      endtxt2 dc.b '0;31;42m LMaustas
te = Diskkopie ',155,'0m
'
948 qs      dc.b 155,'0;31;42m RMau
staste = Ramkopie ',155,
'0m',10
949 Uz0      cerrtxt dc.b 'Fehler in Block $'
950 Ik      bnummer dc.b ' ' . Disk u.U. feh
lerhaft !! Fehlernummer '
951 W5      enummer dc.b ' ',10
952 hL      text ds.l 1
953 4B      textlen ds.l 1
(C) 1988 M&T

```

Listing 2. (Schluß)

Kopieren mit vielen Extras

Jeder Amiga-Fan benötigt ein Programm, das Dateien kopiert. »Supercopy« bietet neben dieser Grundfunktion Features, die bisher nur von kommerziell vertriebenen Programmen, beispielsweise CLI-Mate, bekannt waren.

Überzeugen Sie sich selbst von der Leistungsfähigkeit dieses Tools.

Ein Programm zum komfortablen Kopieren einzelner Dateien gehört in jede Diskettensammlung. Aber ein leistungsfähiges Tool muß heute schon mehr bieten, als nur die Funktion »Kopieren«. Löschen und Umbenennen von Files, Anzeigen der Inhalte von Text- und Bilddateien gehören dazu. Auf dem kommerziellen Sektor sind dies beispielsweise »Zing!«, »Diskmaster« und »CLI-Mate«.

Diese Programme bieten viele leistungsfähige Zusatzfunktionen. Ausführliche Tests dieser Tools fanden Sie im Amiga-Magazin, dort sind die Vor- und Nachteile beschrieben. Alle kommerziellen Tools haben jedoch einen gravierenden Nachteil: sie kosten eine Menge Geld.

»Supercopy« ist eine hervorragende Alternative zu den genannten Programmen. Neben der Grundfunktion Kopieren sind folgende Features enthalten:

- Kopieren mit gleichzeitigem Löschen der Quelldatei (move),
- Kopieren ganzer Directories,
- Umbenennen von Dateien (rename),

Komfort und Leistungsvielfalt

- Löschen von Files,
- Erstellen von Directories (makedir),
- Formatieren von Disketten (mit oder ohne Verify),
- Anzeigen von ASCII-Files (type),
- Anzeigen von Binär-Dateien (hex-type),
- Anzeigen der zu kopierenden Blocks,
- Anzeige des freien Speicherplatzes auf einer Disk (Floppy-, Hard- und RAM-Disk),

- automatische Erkennung der angeschlossenen Geräte,
- Eingabe von CLI-Befehlen ohne Öffnen eines neuen Fensters,
- Anzeige von IFF-Files, wenn ein entsprechender Befehl (show) auf der Diskette ist,
- Disk-backup, wenn sich ein Kopierprogramm (diskcopy) im c-Directory befindet.

Viele Funktionen und leichte Bedienung

Trotz der vielen eingebauten Funktionen ist Supercopy sehr leicht zu bedienen. Wenn Sie das Programm zum ersten Mal starten, sollten Sie allerdings nur mit Kopien Ihrer Disketten arbeiten, um eine mögliche Beschädigung, zum Beispiel durch den Befehl »FORMAT«, zu vermeiden.

Das Programm ist in der Sprache C geschrieben. Haben Sie die Listings 1 bis 8 mit dem Checksummer (Seite 159) eingegeben, sind die Programme mit dem Aztec-C V3.6 zu compilieren und anschließend zu linken. Die nötigen Anweisungen lauten:

Compiler-Anweisungen:

Als erstes ist »includes.c« (Listing 1) zu compilieren:

```
cc -H supercopy.pre includes.c
```

Danach übersetzen Sie die Listings 2 bis 8:

```
sc0.c, sc1.c, sc2.c, sc4.c, sc5.c und sc6.c:  
cc -B -s -I supercopy.pre -o scx.o scx.c  
(X = Name der Datei)  
sc3.c:  
cc -s -I supercopy.pre -o sc3.o sc3.c
```

Linken:

```
ln +g +Cd sc0.o sc1.o  
sc1.o sc3.o sc4.o sc5.o  
sc6.o -o supercopy -lc
```

Das klingt sehr viel komplizierter als es wirklich ist. Auf der Programmservice-Diskette finden Sie das Programm natürlich in lauffähiger Form vor. Hier entfallen lästige Vorarbeiten.

Haben Sie alles richtig gemacht, liegt das Programm als »supercopy« in lauffähiger Form vor. Starten Sie das Programm vom CLI aus mit dem Aufruf

```
supercopy <RETURN>
```

Das Tool wird nun geladen, anschließend erscheint der Arbeitsbildschirm (Bild 1). Jetzt können Sie die gewünschten Manipulationen nach Herzenslust vornehmen.

Die Gliederung des Arbeitsbildschirms

Der Bildschirm zeigt anfangs zwei große leere Fenster links und rechts. Hier erscheinen nach Ihrer Wahl die Inhalte der Speichermedien, zum Beispiel der Diskette in Laufwerk df0:

Die angeschlossenen Geräte sehen Sie im Fenster in der Mitte des Bildschirms. Sollten mehr als vier Geräte vorhanden sein (z.B. drei Floppy-laufwerke, das RAM und eine Harddisk), verschieben Sie mit dem vertikalen Balken rechts neben der Anzeige den Fensterausschnitt.

Klicken Sie eine der Bezeichnungen mit der linken Maustaste an, erscheinen im aktivierten Fenster (links oder rechts) die Dateien. »Aktiv« ist immer die Seite (SOURCE oder DESTINATION), deren

nach unten zeigende Pfeile sichtbar sind.

Es werden immer die ersten dreizehn Einträge angezeigt. Sollten mehr vorhanden sein, sehen Sie diese nach Verschieben des Balkens rechts neben dem Ausgabefenster (bei gedrückter linker Maustaste nach unten ziehen).

Angezeigt wird anfangs das Haupt-Directory (ROOT). In ein Unterverzeichnis wechseln Sie, indem Sie das entsprechende Verzeichnis mit der rechten Maustaste anklicken.

Als Zusatzinformation sehen Sie im Kasten rechts unter der Dateianzeige den freien Speicherplatz auf dem Datenträger.

Die Wurzeln im Haupt-Directory

Der Inhalt eines zweiten Speichermediums, beispielsweise des RAM, läßt sich auf die gleiche Weise im zweiten Ausgabefenster darstellen. Klicken Sie dazu mit der linken Maustaste »DESTINATION« an und anschließend das gewünschte Speichermedium. Sie können auch hier in ein beliebiges Unterverzeichnis wechseln. Es ist auch möglich, in beiden Fenstern verschiedene Verzeichnisse der gleichen Diskette darzustellen.

Meldungen des Programms

Links neben dem waagerechten Balken unten sehen Sie die Bezeichnung »Message«. Hier erscheinen während der Arbeit mit Supercopy häufig Meldungen des Programms.

Die Schalter in der Mitte des Bildschirms besitzen von oben nach unten folgende Bedeutung:

QUIT: beendet die Arbeit mit Supercopy (Rücksprung ins CLI).

PARENT: zeigt den Inhalt des in dem waagerechten Balken oben dargestellten Verzeichnisses an.

ROOT: Wechselt immer in das Hauptverzeichnis.

ALL: Mit diesem Schalter kennzeichnen Sie alle Dateien des aktiven Verzeichnisses. Diese Funktion nimmt Ihnen die Arbeit ab, alle Files mit der linken Maustaste der Reihe nach anzuklicken. Gekennzeichnete Files erkennen Sie an der reversen Darstellung. Um die Wahl einzelner Files rückgängig zu machen, klicken Sie diese mit der linken Maustaste erneut an.

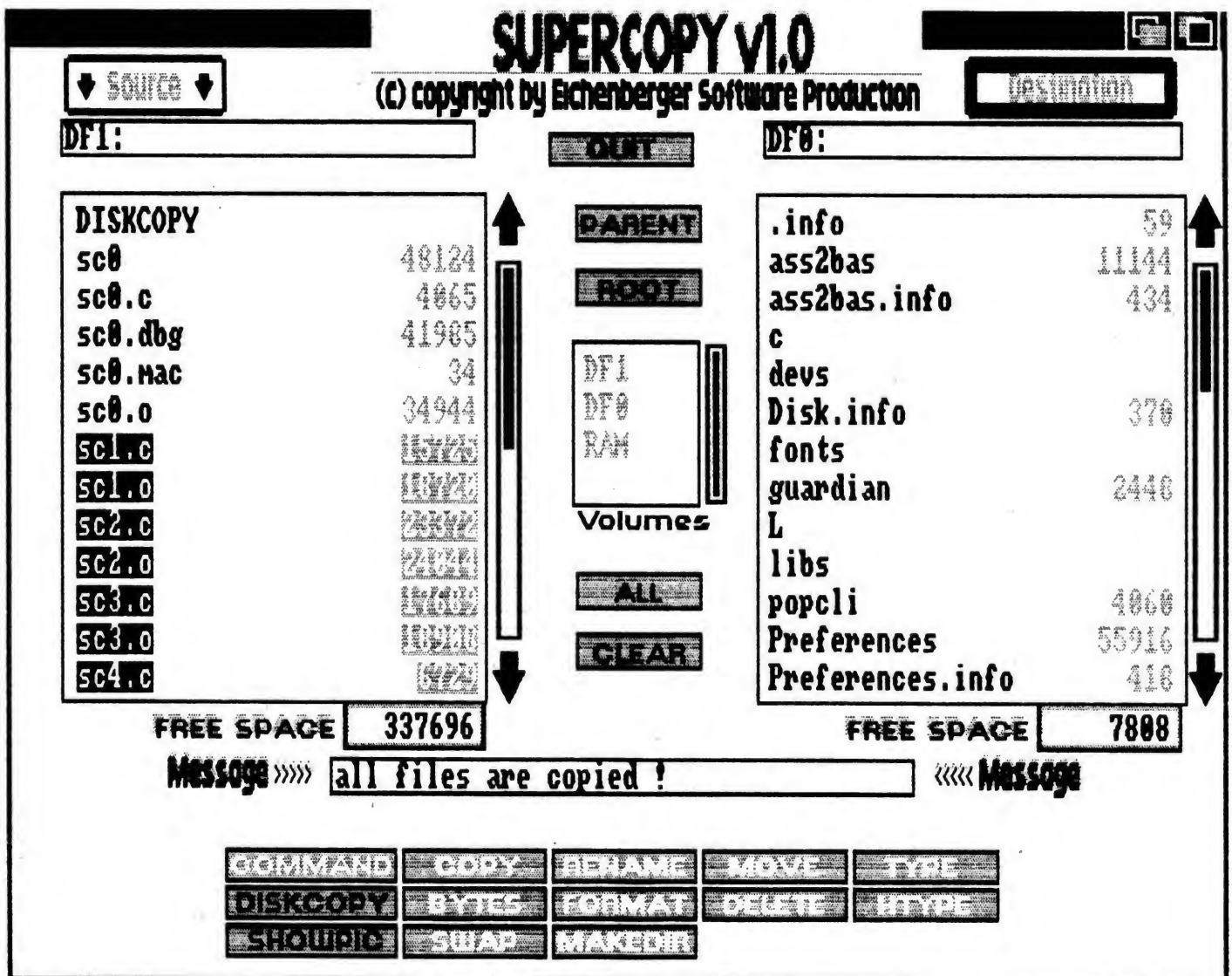


Bild 1. Der Arbeitsbildschirm von »Supercopy« läßt die Funktionsvielfalt des Programms bereits erahnen

CLEAR: hebt die Auswahl und damit die Reversdarstellung aller Dateien im aktiven Verzeichnis auf.

Nachdem wir erfahren haben, welche Möglichkeiten zur Auswahl der Files bestehen, jetzt zu den zahlreichen Funktionen von Supercopy. Im unteren Bildschirmteil erkennen Sie dreizehn Gadgets. Diese tragen beispielsweise die Bezeichnungen »COMMAND« und »COPY«. Folgende Fähigkeiten verbergen sich hinter diesen Schaltern:

COMMAND: Nach Anklicken dieses Schalters mit der linken Maustaste erscheint im Zentrum des Bildschirms ein neues Fenster. Hier sind beliebige CLI-Kommandos einzugeben. Tippen Sie das Kommando (z. B. LIST) ein und bestätigen mit <RETURN>. Statt dessen ist auch ein bereits beim Öffnen des Fensters angezeigter Befehl mit »OK« (linke Maustaste) zu aktivieren. Mit dem Schalter »CANCEL«

brechen Sie die Kommando-Eingabe ab.

COPY: Kopieren aller im »Source«-Verzeichnis markierten Dateien in das aktive Verzeichnis im Feld »Destination«.

Spaß durch Vielfalt

Haben Sie Unterverzeichnisse gewählt (diese erscheinen in einer anderen Farbe), werden alle darin enthaltenen Dateien kopiert.

RENAME: Nach Anklicken dieses Schalters erscheint die Aufforderung, eine Datei zu markieren (»please select a File.«). Klicken Sie mit der linken Maustaste eine beliebige Datei an, erscheint ein neues Fenster in der Bildschirmmitte. Der Name des gewählten Files wird angezeigt und ist beliebig zu ändern. Mit <RETURN> oder dem Schalter »OK« bestätigen Sie die Namensänderung, mit »CANCEL« brechen Sie ohne Änderung ab.

MOVE: wie »COPY«. Zusätzlich wird hier nach dem Kopieren die Quelldatei gelöscht.

TYPE: zeigt den Inhalt einer Textdatei an. Nach dem Anklicken ist eine Datei zu wählen. Der Inhalt dieser Textdatei wird auf einem neuen Screen seitenweise ausgegeben. Weiterblättern erfolgt mit <SPACE>. Die Textausgabe ist durch Anklicken des Schließ-Gadgets in der linken oberen Ecke abzubrechen, durch erneutes Anklicken des Gadgets schließen Sie den Screen.

Das Programm erkennt, wenn die gewählte Datei keine Textdatei ist. Die Funktion wird daraufhin abgebrochen mit der Meldung »not correct. Use HTYPE!«.

BYTES: zeigt die Anzahl der Bytes aller gewählten, also revers dargestellten Dateien. Sie können so schnell feststellen, ob alle Files überhaupt auf einen Datenträger passen. Vergleichen Sie einfach die Byte-

Zahl mit der Anzeige im Feld »FREE SPACE«.

FORMAT: Formatieren einer neuen Diskette. Es erscheint ein neues Fenster, in dem Sie die Bezeichnung des Laufwerkes (df0:, df1:, df2:) und den Diskettenamen angeben. Wählen Sie zuerst das Laufwerk durch Anklicken der gewünschten Bezeichnung mit der linken Maustaste. Bewegen Sie den Mauszeiger anschließend in das Feld unter der Bezeichnung »DISKNAME« und klicken es an. Geben Sie nun den Namen der zu formatierenden Diskette über die Tastatur ein und drücken <RETURN>. Anschließend startet der Formatierungsvorgang, die Spuren werden angezeig.

Wichtig: Supercopy arbeitet im Test beim »Format« fehlerhaft, wenn die Diskettenstation df2: formatiert werden sollte, df1: aber nicht vorhanden war. In diesem Fall steht nur Laufwerk df0: zum Formatieren zur Verfügung.

DELETE: löscht alle selektierten (revers dargestellten) Dateien im aktiven Verzeichnis.

HTYPE: zeigt den Inhalt einer Binär-Datei. Blättern und Abbrechen wie bei »TYPE«.

SWAP: tauscht den Inhalt der Fenster »Source« und »Destination«.

MAKEDIR: legt im aktiven Directory ein neues Unterverzeichnis an. Geben Sie den Namen über die Tastatur ein.

Nützliche Optionen

Supercopy zeigt zwei Schalter in der linken unteren Ecke an. **Die Funktionen »DISKCO-**

PY« und »SHOWPIC« sind nicht im Programm selbst enthalten, sondern erfordern Zusätze. Dies wurde bewußt auf diesem Wege gelöst, um das Listing in einer vertretbaren Länge zu halten.

DISKCOPY: erfordert im Unterverzeichnis »DISKCOPY« ein Kopierprogramm mit dem gleichen Namen. Auf der Programmservice-Diskette befindet sich allerdings ein Kopierprogramm mit diesem Namen.

SHOWPIC: dient der Anzeige von IFF-Grafiken. Im c-Direktory muß sich zu diesem Zweck ein Programm namens »SHOW« befinden.

Wie schon oben erwähnt sind beide Befehle nicht im Programm enthalten, die Hilfs-

routinen sind also vom Anwender eventuell hinzuzufügen.

Zugaben

Mit den Funktionstasten sind noch einige zusätzliche Programmpunkte aufzurufen: <F1> bis <F3>: Verschiedene Farben sind für den Bildschirm zu wählen.

<F4>: Rückkehr zu den normalen Workbench-Farben.

<F5>: Der Arbeitsbildschirm wird um zirka zehn Prozent verkleinert.

<F6>: Rückkehr zur Originalgröße.

<F7>: Erzeugt ein script-File mit der Bezeichnung »script.asc« im aktuellen Verzeichnis. In dieses File werden

alle Directories und Files eingetragen, die selektiert wurden. Bei jedem erneuten Druck auf <F7> werden die gekennzeichneten Files an die bereits bestehende List angehängt. Diese Funktion ist vor allem sinnvoll, wenn Sie eine Übersicht über Ihre Programme und alle Dateien behalten wollen. Mit einem Disksorter können dann diese Files aus der Liste in »script.asc« bequem weiterverarbeitet werden.

Nun sind wir am Ende der Beschreibung aller Befehle des Supercopy angelangt. Wir wünschen Ihnen viel Erfolg beim Austesten der vielen praktischen Funktionen dieses tollen Tools.

(R. Eichenberger/D. Körtke/rs)

Programmname: includes.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Compiler: Aztec V3.6a

Bemerkungen: Bitte beachten Sie die Compiler- und Linker-Anweisungen im Text

Programmname: includes.c

```
1 Pe0 #include <intuition/intuition.h>
2 J7 #include <intuition/intuitionbase.h>
3 60 #include <exec/types.h>
4 BG #include <exec/memory.h>
5 3q #include <libraries/dos.h>
6 zx #include <libraries/dosextens.h>
7 GR #include <stdio.h>
(C) 1988 M&T
```

Listing 1. »includes.c« ist als erstes Programm zu compilieren. Bitte geben Sie das Listing mit dem Checksummer (Seite 159) ein.

Programmname: sc0.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Bemerkung: siehe Listing 1

Programmname: sc0.c

```
1 Nm0 /******
*****
2 Sc3 Supercopy v1.0 (w) in 1988 by Eichenberger Software Prod
uction
3 Vq0 Last UpDate:
4 SD Wed Oct 5 14:33:35 1988
5 Wv *****
*****
6 qF /*=====
7 tU4 HEADERS
8 IA1 =====*/
9 Vy0 #include <functions.h>
```

```
10 ds #include <:sc_struct.h>
11 sD #include <drawbox.h>
12 a3 char dummy[1];
13 BD struct Source src;
14 9m struct Destination dst;
15 x1 struct Volumes vlm;
16 jC struct IntuitionBase *IntuitionBase;
17 3g struct GfxBase *GfxBase;
18 E7 struct Window *Main_Window;
19 iv struct Window *Rename_Window;
20 Np struct Window *Format_Window;
21 10 struct RastPort *mainRP;
22 HN struct Gadget *sgad[13]; /* SourceBlock Gadgets */
23 DD struct Gadget *dgad[13]; /* DestinationBlock Gadgets */
24 1Z struct Gadget *vgad[8]; /* VolumesBlock Gadgets */
25 Ds #include <vd0:supercopy.h>
26 WL main(){
27 Hn2 OpenLibraries();
28 6u InitWindow();
29 zA3 MemoryAlloc();
30 g0 SaveDefaultColors();
31 Wz Record_Volumes();
32 kv2 MAIN_WaitMsg();
33 Hg0 /*=====
34 MB4 Oeffnen der Libraries
35 jb1 =====*/
36 U20 OpenLibraries(){
37 7P2 if(!(IntuitionBase = (struct IntuitionBase *)OpenLibrary(
"intuition.library",33L))){
38 td puts("can't open intuition.library");
39 hB2 if(!(GfxBase = (struct GfxBase *)OpenLibrary("graph
ics.library",33L))){
40 eu4 puts("can't open graphics.library");
41 Po0 /*=====
42 uv2 Schliessen der Libraries
43 rj1 =====*/
44 ty0 CloseLibraries(){
45 9f2 CloseLibrary(GfxBase);
46 1a CloseLibrary(IntuitionBase);
47 Vu0 /*=====
48 5v3 Init. des Hauptwindows
49 xp1 =====*/
50 aQ0 InitWindow(){
51 ba2 InitBlockGadgets();
52 RP if(!(Main_Window = OpenWindow(&MAIN_window))){
53 6W4 puts("can't open Mainwindow");
54 Hd2 mainRP=Main_Window->RPort;
55 1I DrawBorder(mainRP,&MAIN_BorderList1,OL,OL);
56 W5 DrawImage (mainRP,&MAIN_ImageList1 ,OL,OL);
57 Uc DrawImage (mainRP,&MAIN_Image36 , 77L,186L);
58 dB DrawImage (mainRP,&MAIN_Image36 ,441L,186L);
```

Listing 2. »sc0.c« bitte mit dem Checksummer eingeben


```

59 Va3 SetAPen(mainRP,1L);
60 Ip drawbox (Main_Window->RPort,177L,183L,251L,194L);
61 Fk drawbox (Main_Window->RPort,178L,182L,250L,193L);
62 ag SetAPen(mainRP,2L);
63 xH drawbox (Main_Window->RPort,179L,183L,249L,193L);
64 af SetAPen(mainRP,1L);
65 hM drawbox (Main_Window->RPort,541L,183L,615L,194L);
66 gK drawbox (Main_Window->RPort,542L,182L,615L,193L);
67 fL SetAPen(mainRP,2L);
68 kP drawbox (Main_Window->RPort,543L,183L,614L,193L);
69 ID AddGLList (Main_Window,&MAIN_clear,5L,NULL);
70 JZ RefreshGLList(&MAIN_clear,Main_Window,NULL,5L);
71 tIO /*=====
72 uR3 Init. der BlockGadgets
73 LD1 =====*/
74 bJO InitBlockGadgets(){
75 lJ2 register UBYTE i;
76 KJ for(i=0;i<=12;i++){
77 Yf4 sgad[i]=AllocMem((long)sizeof(struct Gadget),MEMF_CLEAR
] MEMF_CHIP);
78 Rh CopyGadgets(&MAIN_firstSBlock,sgad[i]);
79 G1 sgad[i]->TopEdge = MAIN_firstSBlock.TopEdge+i*10;
80 UR sgad[i]->GadgetID = i+3;
81 Po2 for(i=0;i<=12;i++){
82 914 dgad[i]=AllocMem((long)sizeof(struct Gadget),MEMF_CLEAR
] MEMF_CHIP);
83 pb CopyGadgets(&MAIN_firstDBlock,dgad[i]);
84 Pd dgad[i]->TopEdge = MAIN_firstDBlock.TopEdge+i*10;
85 WS dgad[i]->GadgetID = i+48;
86 js2 for(i=0;i<=3;i++){
87 oy4 vgad[i]=AllocMem((long)sizeof(struct Gadget),MEMF_CLEAR
] MEMF_CHIP);

```

```

88 wI CopyGadgets(&MAIN_firstVBlock,vgad[i]);
89 oA vgad[i]->TopEdge = MAIN_firstVBlock.TopEdge+i*10;
90 4G vgad[i]->GadgetID = i+19;
91 Vt3 for(i=0;i<=11;i++){
92 4I6 sgad[i]->NextGadget = sgad[i+1];
93 dN dgad[i]->NextGadget = dgad[i+1];
94 oK if(i <= 6) vgad[i]->NextGadget = vgad[i+1];
95 uM2 sgad[12]->NextGadget = dgad[0];
96 BE dgad[12]->NextGadget = vgad[0];
97 gU vgad[7]->NextGadget = NULL;
98 pC MAIN_hType.NextGadget= sgad[0];
99 qW0 MAIN_cleanup(){
100 2G3 register ULONG i;
101 j82 for(i=0;i<=12;i++){
102 5v6 FreeMem(sgad[i],(long)sizeof(struct Gadget));
103 tU FreeMem(dgad[i],(long)sizeof(struct Gadget));
104 6X if(i <= 3) FreeMem(vgad[i],(long)sizeof(struct Gadget));
105 L03 MAIN_hType.NextGadget=NULL;
106 kX MAIN_MAIN_d_pathSInfo.Buffer = (UBYTE *)&dummy;
107 FH MAIN_MAIN_s_pathSInfo.Buffer = (UBYTE *)&dummy;
108 Q52 if(Main_Window) CloseWindow(Main_Window);
109 qt CloseLibraries();
110 aS0 CopyGadgets(source,destination)
111 dL3 BYTE *source,*destination;
112 ES register ULONG i;
113 g7 for(i=0;i<sizeof(struct Gadget);i++){
114 Ye6 *destination++=*source++;
(C) 1988 M&T

```

Listing 2. (Schluß)

Programmname: sc1.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Bemerkung: siehe Listing 1

Programmname: sc1.c

```

1 Nm0 /*=====
*****
2 Sc3 Supercopy v1.0 (w) in 1988 by Eichenberger Software Prod
uction
3 CYK Funktionen fuer die Directoryverwaltung
4 Wr0 Last UpDate:
5 zt Fri Apr 29 20:48:29 1988
6 st *****
*****SC1*/
7 rG /*=====
8 uV4 HEADERS
9 JB1 =====*/
10 rS0 #include<exec/types.h>
11 XO #include<functions.h>
12 fu #include<:sc_struct.h>
13 xM /*=====
14 3x4 DEFINES
15 PH1 =====*/
16 jKO #define UP 100
17 bL #define DOWN 101
18 NO #define SOURCE OL
19 z9 #define DESTINATION 1L
20 lJ #define strlen _BUILTIN_strlen
21 Go #define strcpy _BUILTIN_strcpy
22 F8 #define strcmp _BUILTIN_strcmp
23 7W /*=====
24 eG4 GLOBALE VARIABLEN
25 ZR1 =====*/
26 Of0 extern LONG Wait_on_Message();
27 7b extern struct Source src;
28 Fv extern struct Destination dst;
29 M8 extern struct Volumes vlm;
30 LW extern struct GfxBase *GfxBase;
31 xp extern struct RastPort *mainRP;
32 a9 extern struct Window *Main_Window;
33 OJ extern struct StringInfo MAIN_MAIN_s_pathSInfo;

```

```

34 Xb extern struct StringInfo MAIN_MAIN_d_pathSInfo;
35 ng extern struct Gadget MAIN_source,MAIN_destination,MAIN_s_pa
th,MAIN_d_path,
36 mIJ *sgad[13],*dgad[13],*vgad[13];
37 xM0 /*=====
*****
38 8QN vergleicht 2 Strings (fuer qsort)
39 4T0 *****
*****/
40 nk cmp(a,b)
41 JV3 char **a,**b;
42 eB register int i;
43 4u char str1[180],str2[180],*lcase();
44 KQ lcase(str1,*a);
45 VY lcase(str2,*b);
46 lW return(strcmp(str1,str2));
47 Aw0 char *lcase(to, from)
48 l53 char *to;
49 Ry char *from;
50 Se4 char *temp = to;
51 6P while (*from)
52 NI8 if (*from < 'A' ] ] *from > 'Z')
53 XeC *to++ = *from++;
54 hU8 else
55 mqC *to++ = (*from++ + ' ');
56 FU4 *to = '\0';
57 Dq return(temp);
58 Ih0 /*=====
*****
59 lU9 Liest das Directory in den Source- oder Destinati
on Buffer ein
60 hR0 Sat Apr 30 21:00:46 1988
61 Qp *****
*****/
62 NA Read_Directory(path,flag)
63 PU3 char path[];
64 fT2 ULONG flag;
65 28 register int i=0;
66 9r struct FileLock *altlock;
67 ZK struct FileLock *neulock; /* Zeiger auf Lock-Structure */
68 Xu LONG success;
69 2T neulock = Lock(path,ACCESS_READ); /* ermittelt LockStru
cture */

```

Listing 3. »sc1.c« bitte mit dem Checksummer eingeben

```

70 w6 if (neulock != 0)
71 S76 altlock = CurrentDir(neulock);
72 UD verfolgepfad(neulock,flag,altlock); /* ver
73 Em folge Pfad */
74 1o2 altlock = CurrentDir(altlock);
75 YP5 else
76 Hs6 WriteMessage("can't lock selected dir");
77 4R3 return(-1);
78 660 Full_Update(flag);
79 81 struct FileLock *lock;
80 vJ ULONG flag;
81 O6 struct FileLock *altlock;
82 3F3 LONG Wait_on_Message();
83 gC2 register WORD i;
84 pK3 struct FileInfoBlock *m;
85 9I struct FileLock *newlock,*oldlock,*ignoredlock;
86 M1 LONG gadid;
87 Dm WORD success;
88 w7 if(!lock) return(0);
89 Td m = (struct FileInfoBlock *)
90 8Q9 AllocMem((long)sizeof(struct FileInfoBlock),MEMF_C
LEAR) MEMF_PUBLIC);
91 He3 success = Examine(lock,m);
92 Wa2 while (success != 0)
93 cg6 success = ExNext(lock,m); /* examine the next ent
ry */
94 y8 if(success){
95 dn8 switch(flag){
96 zZ case SOURCE:
97 KqK src.list.entries++;
98 qJ strepy(src.list.name[src.list.entries-1]
+1,
m->fib_FileName);
99 MxR src.list.length[src.list.entries-1] = m-
100 KAK >fib_Size;
101 kz if(m->fib_DirEntryType > 0){
102 dnM src.list.dir[src.list.entries-1] = 1;
103 UHK else
104 ZnM src.list.dir[src.list.entries-1] = 0;
105 cLK break;
106 N38 case DESTINATION:
107 9IK dst.list.entries++;
108 Tf strepy(dst.list.name[dst.list.entries-1]
+1,
m->fib_FileName);
109 W7R dst.list.length[dst.list.entries-1] = m-
110 rnK >fib_Size;
111 u9 if(m->fib_DirEntryType > 0){
112 t4M dst.list.dir[dst.list.entries-1] = 1;
113 eRK else
114 p4M dst.list.dir[dst.list.entries-1] = 0;
115 mvK break;
116 Qv8 }/* end of switch */
117 OrA }/* end of if */
118 506 }/* end of while */
119 Mg3 if (lock) UnLock(lock);
120 07 FreeMem(m,(long)sizeof(struct FileInfoBlock));
121 2y0 Full_Update(flag)
122 Ft3 LONG flag;
123 3D2 if((src.list.entries > 1) && (flag == SOURCE)) Sort_Files
(flag);
124 hS if((dst.list.entries > 1) && (flag == DESTINATION)) Sort_
Files(flag);
125 zz3 Write_Info(flag,2,0);
126 2T DisplayFiles(flag);
127 8H PropGad(flag,0);
128 EO PropGad(flag,1);
129 Rq0 /*****
*****
reserviert Speicher
200 Filenamen a 26 Zeichen
180 Zeichen fuer den aktuellen Pfad
133 f10 Sat May 28 14:33:22 1988
134 b0 *****
*****
MemoryAlloc(){
135 r4 register ULONG i;
136 cq for(i=0;i<200;i++){
137 mx2 src.list.name[i] = (char *)AllocMem(27L,MEMF_CLEAR);
138 1Y4 src.spath = (char *)AllocMem(181L,MEMF_CLEAR);
139 bx2 MAIN_MAIN_s_pathSInfo.Buffer = (UBYTE *)src.spath;
140 f13 for(i=0;i<200;i++){

```

```

142 UK4 dst.list.name[i] = (char *)AllocMem(27L,MEMF_CLEAR);
143 sF2 dst.dpath = (char *)AllocMem(181L,MEMF_CLEAR);
144 SH3 MAIN_MAIN_d_pathSInfo.Buffer = (UBYTE *)dst.dpath;
145 D92 for(i=0;i<30;i++){
146 Op4 vlm.n[i] = (char *)AllocMem(7L,MEMF_CLEAR);
147 j80 /*****
*****
148 K6I Vorheriges Directory im Pfad aktuell mach
en
149 WIO Sun Apr 24 18:16:16 1988
150 rG *****
*****
151 zQ Parent_Dir(pfad,flag)
152 XK2 char pfad[];
153 6u ULONG flag;
154 bd ULONG count;
155 G1 UBYTE depth;
156 1H count=strlen(pfad)-1;
157 dn switch(flag){
158 gH4 case SOURCE : depth = src.depth;break;
159 dN case DESTINATION: depth = dst.depth;break;
160 fJ default: depth = flag;
161 rL3 if(depth == 0) return(-1);
162 fa2 if (depth>1){
163 Tm4 while (pfad[count] != '/'){
164 Tw7 pfad[count]=0;
165 L1 count--;
166 Vy4 pfad[count]=0;
167 WJ2 else
168 U54 while ((pfad[count] != ':') ] (count==0)){
169 Y16 pfad[count]=0;
170 Qq count--;
171 z32 depth--;
172 s2 switch(flag){
173 Jr4 case SOURCE : src.depth=depth;break;
174 zb case DESTINATION: dst.depth=depth;break;
175 1M default: break;
176 Cb0 /*****
*****
177 XOG eingegebener Pfad im StringGadget aktuell m
achen
178 9AV gibt depth zurueck
179 jUO Sun Apr 24 19:01:15 1988
180 Lk *****
*****
181 mf UBYTE NewPath(pfad)
182 1o2 char pfad[];
183 vu register WORD count;
184 O3 register UBYTE depth=0;
185 K6 count = strlen(pfad);
186 hS while(count > 0){
187 c24 while((pfad[--count] != '/') && (count > 0))
188 Hg {;
189 77 depth++;
190 l86 if(pfad[strlen(pfad)-1] == ':'){
191 p19 depth = 0;
192 qF2 return(depth);
193 Ts0 /*****
*****
194 4AI ein neues Directory an einen Pfad anhaeng
en
195 6sO Sat Apr 30 21:17:54 1988
196 b0 *****
*****
197 pr PathCat(newpath,flag)
198 Oa3 char newpath[];
199 qe ULONG flag;
200 KU switch(flag){
201 gG5 case SOURCE:
202 C3C if(src.depth > 0) strcat(src.spath,"/");
203 OP src.depth++;
204 BD strcat(src.spath,newpath);
205 oQ RefreshGList(&MAIN_s_path,Main_Window,NULL,1L);
206 FO break;
207 Og5 case DESTINATION:
208 wMC if(dst.depth > 0) strcat(dst.dpath,"/");
209 lD8 dst.depth++;
210 JmC strcat(dst.dpath,newpath);
211 S2 RefreshGList(&MAIN_d_path,Main_Window,NULL,1L);
212 LU break;
213 nCO /*****
*****
214 f5J alle reservierten Strukturen zurueckgebe
n

```



```

215 n00 Sun Apr 24 20:15:40 1988
216 vK *****/
217 bX MEM_cleanup(){
218 wA3 register ULONG i;
219 6H2 for(i=0;i<200;i++){
220 GD4 if(src.list.name[i]) FreeMem(src.list.name[i],27L);
221 5k2 if(src.spath) FreeMem(src.spath,181L);
222 9K for(i=0;i<200;i++){
223 9K4 if(dst.list.name[i]) FreeMem(dst.list.name[i],27L);
224 dS2 if(dst.dpath) FreeMem(dst.dpath,181L);
225 VR for(i=0;i<30;i++){
226 Jo4 FreeMem(vlm.n[i],7L);
227 lQ0 /*****
*****
228 yAK Anzeigen der Files in einer der Bloecke
229 cJO Sun Apr 24 20:13:39 1988
230 9Y *****/
*****/
231 YY DisplayFiles(flag)
232 NB3 ULONG flag;
233 JR register UWORD xStart,yStart;
234 p5 register struct RastPort *rp;
235 a0 register UWORD i;
236 HM2 char length[10];
237 Lx3 register ULONG count;
238 gM register char *name;
239 2p rp = mainRP;
240 y8 switch(flag){
241 Ku6 case SOURCE:
242 hy9 xStart = 37; yStart=58;
243 lQ for(i=0;i<=12;i++){
244 DwC ChangeColors();
245 pR count = i+src.scroll_count;
246 ZK name = src.list.name[count];
247 CS8 if(src.list.dir[count] == 1) SetAPen(rp,3L);
248 KF if(src.list.select[count] == 1){
249 b3F SetDrMd(rp,INVERSVID|JAM2);
250 oHC if(strlen(name) <= 20){
251 Y3F Move(rp,(long)xStart,(long)yStart+i*10);
252 mP Text(rp,name,
253 oqG (long)strlen(name));
254 JeC else{
255 c7F Move(rp,(long)xStart,(long)yStart+i*10);
256 qT Text(rp,name,
257 gLG 20L);
258 GK0 if(((count) < src.list.entrys) &&
259 lOG (src.list.dir[count] == 0)){
260 f8F sprintf(length,"%ld",src.list.length[count])
;
261 EDC SetAPen(rp,2L);
262 ZyF Move(rp,(long)248-TextLength(rp,length,(long)
strlen(length)),
263 k7I (long)yStart+i*10);
264 nUF Text(rp,length,(long)strlen(length));
265 Cl6 break;
266 xd case DESTINATION:
267 kc9 xStart = 401; yStart=58;
268 Qp for(i=0;i<=12;i++){
269 cLC ChangeColors();
270 tY count = i+dst.scroll_count;
271 JA9 name = dst.list.name[count];
272 2G8 if(dst.list.dir[count] == 1) SetAPen(rp,3L);
273 A3 if(dst.list.select[count] == 1){
274 OSF SetDrMd(rp,INVERSVID|JAM2);
275 DgC if(strlen(name) <= 20){
276 xSF Move(rp,(long)xStart,(long)yStart+i*10);
277 Bo Text(rp,name,
278 DFG (long)strlen(name));
279 83C else{
280 lWF Move(rp,(long)xStart,(long)yStart+i*10);
281 Fs Text(rp,name,
282 5kG 20L);
283 4OC if(((count) < dst.list.entrys) &&
284 8YG (dst.list.dir[count] == 0)){
285 mIF sprintf(length,"%ld",dst.list.length[count])
;
286 xx Move(rp,(long)612-TextLength(rp,length,(long)
strlen(length)),
287 8V (long)yStart+i*10);
288 feC SetAPen(rp,2L);
289 CfF Text(rp,length,(long)strlen(length));
290 bk6 break;

```

```

291 Fk3 }/* end of switch */
292 wM0 }/* end of block */
293 5U /*****
*****
294 3aK Anzeigen eines Files in einer der Bloec
ke
295 fQ0 Sat Apr 30 15:16:42 1988
296 Dc *****
*****/
297 ay DisplaySingle(flag,direction)
298 RF3 ULONG flag;
299 CB ULONG direction;
300 Rx register ULONG xStart,yStart;
301 uA register struct RastPort *rp;
302 f5 register UWORD i;
303 MR2 char length[10];
304 Q23 register ULONG count;
305 lR register char *name;
306 7u rp = mainRP;
307 3D switch(flag){
308 Pz6 case SOURCE:
309 VY9 xStart = 37; yStart=178; i=12;
310 OC6 if(direction == UP) {i=0;yStart=58;}
311 tV9 count = i+src.scroll_count;
312 dO name = src.list.name[count];
313 K36 ChangeColors();
314 HX if(src.list.dir[count] == 1) SetAPen(rp,3L);
315 PK if(src.list.select[count] == 1){
316 g8C SetDrMd(rp,INVERSVID|JAM2);
317 tM9 if(strlen(name) <= 20){
318 VqC Move(rp,(long)xStart,(long)yStart);
319 rU Text(rp,name,
320 tvD (long)strlen(name));
321 oj9 else{
322 ZuC Move(rp,(long)xStart,(long)yStart);
323 vY Text(rp,name,
324 lQD 20L);
325 LP9 if(((count) < src.list.entrys) &&
326 6TD (src.list.dir[count] == 0)){
327 KDC sprintf(length,"%ld",src.list.length[count]);
328 JI9 SetAPen(rp,2L);
329 e3C Move(rp,(long)248-TextLength(rp,length,(long)st
rlen(length)),
330 w4G (long)yStart);
331 sZC Text(rp,length,(long)strlen(length));
332 HQ6 break;
333 2I case DESTINATION:
334 KW9 xStart = 401; yStart=178; i=12;
335 Pb6 if(direction == UP) {i=0;yStart=58;}
336 xc9 count = i+dst.scroll_count;
337 nE name = dst.list.name[count];
338 JS ChangeColors();
339 7L6 if(dst.list.dir[count] == 1) SetAPen(rp,3L);
340 F8 if(dst.list.select[count] == 1){
341 5XC SetDrMd(rp,INVERSVID|JAM2);
342 lI9 if(strlen(name) <= 20){
343 uFC Move(rp,(long)xStart,(long)yStart);
344 Gt Text(rp,name,
345 IKD (long)strlen(name));
346 DB9 else{
347 yJC Move(rp,(long)xStart,(long)yStart);
348 Kx Text(rp,name,
349 ApD 20L);
350 9T9 if(((count) < dst.list.entrys) &&
351 DdD (dst.list.dir[count] == 0)){
352 hg9 SetAPen(rp,2L);
353 sOC sprintf(length,"%ld",dst.list.length[count]);
354 33 Move(rp,(long)612-TextLength(rp,length,(long)st
rlen(length)),
355 LTF (long)yStart);
356 HyC Text(rp,length,(long)strlen(length));
357 gp6 break;
358 Kp3 }/* end of switch */
359 lR0 }/* end of block */
360 Hs ChangeColors(){
361 s82 register struct RastPort *rp;
362 lo3 rp = mainRP;
363 rp6 SetBPen(rp,0L);
364 qo SetAPen(rp,1L);
365 GV SetDrMd(rp,JAM2);
366 GfO /*****
*****

```

Listing 3. (Fortsetzung)

```

367 LwJ                alle selektierten Filenames-Flags loesch
                        en
368 qV0 Sat Apr 30 21:26:25 1988
369 On *****
*****/
370 NC ClearAll(flag)
371 cQ3     ULONG flag;
372 Qe     register ULONG i;
373 7H2     switch(flag){
374 T36         case SOURCE:
375 Jb9             for(i=0;i<=199;i++){
376 3uD                 src.list.select[i] = 0;
377 Jg                     *src.list.name[i] = 0;
378 qG                     src.list.length[i] = 0;
379 ha9                 src.list.entrys=0;
380 4u                     src.scroll_count=0;
381 fJ                     SetAPen(mainRP,OL);
382 JA                     EreaseBlock(SOURCE);
383 6F                     break;
384 rX6         case DESTINATION:
385 T19             for(i=0;i<=199;i++){
386 smD                 dst.list.select[i] = 0;
387 Bb                     *dst.list.name[i] = 0;
388 f8                     dst.list.length[i] = 0;
389 WS9                 dst.list.entrys=0;
390 tm                     dst.scroll_count=0;
391 pt                     SetAPen(mainRP,OL);
392 7t                     EreaseBlock(DESTINATION);
393 GP                     break;
394 170 /*****
*****
395 roT                Loescht ein Block-Feld
396 K80 Sun May 01 13:44:48 1988
397 qF *****
*****/
398 Kc ClearBlock(flag,count)
399 4s3     ULONG flag;
400 8b     UWORD count;
401 z3     SetAPen(mainRP,OL);
402 vp     SetDrMd(mainRP,JAM2);
403 b1     switch(flag){
404 xX6         case SOURCE:
405 WyB             RectFill(mainRP,37L,(long)count*10+52,250L,(long)
count*10+59);
406 TeC             break;
407 Eu6         case DESTINATION:
408 gNB             RectFill(mainRP,401L,(long)count*10+52,614L,(lon
g)count*10+59);
409 WfC             break;
410 Af3     }/* end of switch */
411 z00 /*****
*****
412 yGM                loeschen eines Blockes mit RectFill
413 n50 Sat May 07 20:16:15 1988
414 7W *****
*****/
415 g2 EreaseBlock(flag)
416 zd3     LONG flag;
417 FJ     SetAPen(mainRP,OL);
418 Xu     SetDrMd(mainRP,OL);
419 r1     switch(flag){
420 Dn6         case SOURCE:
421 ks             RectFill(mainRP,32L,50L,250L,181L);
422 js9             break;
423 UA6         case DESTINATION:
424 f5             RectFill(mainRP,396L,50L,614L,181L);
425 mv9             break;
426 Ed0 /*****
*****
427 no3     Markieren der Files, damit ich nach qsort die alte Pos.
rausfinde
428 Kc     danach
429 M10 *****
*****/
430 mM Sort_Files(flag)
431 a03     ULONG flag;
432 wT     register int i;
433 tZ     register UBYTE count;
434 Fh     char *m;
435 4g2     UBYTE dir[200];
436 tr3     ULONG length[200];
437 9J     switch(flag){
438 V56         case SOURCE:

```

```

439 9P9         for(i=0;i<src.list.entrys;i++){
440 9kC             m=src.list.name[i]+strlen(src.list.name[i]);
441 13             *(m)=(UBYTE)i+1;
442 Ov             *(m+1)=(UBYTE)0;
443 I5             dir[i]=src.list.dir[i];
444 3o             length[i]=src.list.length[i];
445 sk9         qsort(src.list.name,(unsigned)(src.list.entrys),(u
nsigned)4,(ULONG)cmp);
446 GW         for(i=0;i<src.list.entrys;i++){
447 GrC             m=src.list.name[i]+strlen(src.list.name[i]);
448 eS             count = *(m-1);
449 dzF             src.list.dir[i] =dir[--count];
450 kY             src.list.length[i]=length[count];
451 md             *(m-1)=(char)0;
452 DM8             break;
453 ye4         case DESTINATION:
454 3M9             for(i=0;i<dst.list.entrys;i++){
455 rBC                 m=dst.list.name[i]+strlen(dst.list.name[i]);
456 xI                 *(m)=(UBYTE)i+1;
457 FA                 *(m+1)=(UBYTE)0;
458 I1                 dir[i]=dst.list.dir[i];
459 x1                 length[i]=dst.list.length[i];
460 XV9         qsort(dst.list.name,(unsigned)(dst.list.entrys),(u
nsigned)4,(ULONG)cmp);
461 AT         for(i=0;i<dst.list.entrys;i++){
462 yIC             m=dst.list.name[i]+strlen(dst.list.name[i]);
463 th             count = *(m-1);
464 XwF             dst.list.dir[i] =dir[--count];
465 eV             dst.list.length[i]=length[count];
466 1s             *(m-1)=(char)0;
467 Sb8             break;
468 f00 _cli_parse(){
469 Lb _wb_parse(){
(C) 1988 M&T

```

Listing 3. (Schluß)

Programmname: sc2.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Bemerkung: siehe Listing 1

Programmname: sc2.c

```

1 Nm0 /*****
2 Rb5     Supercopy v1.0 (w) in 1988 by Eichenberger Software Prod
uction
3 fK6     alle Gadget-Funktionen. z.B. copy, rename, format usw
4 Lq0 Tue Apr 26 20:07:35 1988
5 xu     *****
*****SC2*/
6 qF     /*=====
7 tU4     HEADERS
8 IA1     =====*/
9 Vy0     #include<functions.h>
10 ds     #include<:sc_struct.h>
11 vK     /*=====
12 1v4     DEFINES
13 NF1     =====*/
14 Jw0     #define SOURCE OL
15 v5     #define DESTINATION 1L
16 OP     /*=====
17 X94     GLOBALE VARIABLEN
18 SK1     =====*/
19 zTO     extern struct Source src;
20 7n     extern struct Destination dst;
21 EO     extern struct Volumes v1m;
22 DO     extern struct GfxBase *GfxBase;
23 ph     extern struct RastPort *mainRP;
24 S1     extern struct Window *Main_Window;
25 Sk     extern struct DosLibrary *DOSBase;
26 vU     extern struct NewWindow rename_nw;
27 q3A     struct Window *Rename_Window;
28 wr0     extern struct Requester REQ_requester;
29 GO     extern UBYTE MessageLine[80];

```



```

30 wa extern UBYTE r_m_buffer[80];
31 St extern UBYTE diskname_buffer[80];
32 VC extern struct Gadget MAIN_message;
33 kC extern struct Gadget rename_r_m;
34 VH extern struct Gadget MAIN_d_path;
35 FG extern struct Gadget MAIN_s_path;
36 JM extern struct StringInfo MAIN_MAIN_s_pathSInfo;
37 ae extern struct StringInfo MAIN_MAIN_d_pathSInfo;
38 pB extern struct Image FRM_Image3;
39 KM extern struct Image FRM_Image8;
40 RP extern struct Image FRM_Image9;
41 nG extern struct Image REQ_Image2;
42 iF extern struct Image REQ_Image1;
43 Bs extern struct Gadget REQ_name;
44 Vu extern struct Gadget FRM_cancel;
45 Gu extern struct Gadget FRM_ok;
46 81 extern struct Image command_img;
47 2k #define FRM_ImageList1 FRM_Image3
48 8X /*****
49 yOI          kopiert ein File von source auf destinati
              on
50 LGJ          0= normal kopi
              eren
51 JL          1= source loes
              chen
52 fC0 Tue Apr 26 19:16:16 1988
53 Ih *****/
54 fg Copy(flag)
55 Jt3  BOOL flag;
56 mz  register ULONG i,t=0;
57 db  struct FileLock *oldlock,*newlock;
58 kJ  char delete[1000];
59 vP  register LONG entrys;
60 fH  entrys = src.list.entrys;
61 67  for(i=0;i<entrys;i++){
62 xW6      if(src.list.select[i] == 1){
63 uZ9          if(src.list.dir[i] == 1){
64 19C              CopyDir(src.list.name[i],(ULONG)0,(BOOL)flag);
65 nA              insert_file_entry(i);
66 CL              t++;
67 id9          else{
68 ENC              t++;
69 Y8              if(CopyFile(src.list.name[i])==-1){
70 BmA                  return(-1);
71 pAC              if(flag == 1){
72 yaF                  namecat(src.list.name[i],delete,SOURCE);
73 jF                  if(DeleteFile(delete) == NULL){
74 DOI                      detect_errors();
75 gd                      goto end;
76 eOC                  Write_Info(DESTINATION,1,src.list.length[i]);
77 zM                  insert_file_entry(i);
78 kd0 end:
79 xI3          if(flag == 1){
80 PQ6              for(i=0;i<entrys;i++){
81 dk9                  if(src.list.select[i] == 1) kill_file_entry(SOURCE
,i--);
82 C73          WriteMessage("all files are copied !");
83 bh          ClearAllSelected(SOURCE);
84 xq          dst.scroll_count=0;
85 30          if(flag == 1){
86 J26              print_files(SOURCE);
87 Yc3          print_files(DESTINATION);
88 t9          if(t==0){
89 P26              WriteMessage("first select files (SOURCE).");
90 nR0 print_files(flag)
91 6u3          ULONG flag;
92 DO          EresaseBlock(flag);
93 ef          Sort_Files(flag);
94 bk          PropGad(flag,0);
95 hr          PropGad(flag,1);
96 Yz          DisplayFiles(flag);
97 XX          Write_Info(flag,2,0);
98 2B0 CopyFile(source)
99 KP3          char source[];
100 aR          register struct FileHandle *fh1,*fh2;
101 dF          char filename[500],destination[500];
102 Q1          char *file_buffer=0;
103 89          LONG error;
104 m8          LONG count,count1;
105 BH          struct FileHandle *Open();
106 Sm          namecat(source,filename,SOURCE);

```

```

107 5Z          namecat(source,destination,DESTINATION);
108 YL          WriteMessage(filename);
109 O1          if((fh1 = Open(filename,MODE_OLDFILE))==0){ /* source n
ame */
110 8H6              puts("can't open");
111 mm              error = IoErr();
112 9J              goto cleanup;
113 Ty3          if((fh2 = Open(destination,MODE_NEWFILE))==0){ /* dst n
ame */
114 CL6              puts("can't open");
115 qq              error = IoErr();
116 Dn              goto cleanup;
117 5S3          file_buffer = (char *)AllocMem(11264L,MEMF_CLEAR);
118 W3          while(count=Read(fh1,file_buffer,5632L)){
119 Wh6              if(Write(fh2,file_buffer,(LONG)count) != count){
120 1T9                  WriteMessage("Write-Error");
121 Vb                  if(file_buffer != NULL) FreeMem(file_buffer,1126
4L);
122 jF                  Close(fh2);
123 fA                  Close(fh1);
124 3e                  return(-1);
125 1O3          if(count == -1){
126 iI6              WriteMessage("Read-Error");
127 Qw0 cleanup:
128 ci3          if(file_buffer != NULL) FreeMem(file_buffer,11264L);
129 qM          Close(fh2);
130 mH          Close(fh1);
131 GL0 } /* end of block */
132 Ut          /*****
133 sxV          Requester zeichnen
134 b00 *****/
135 j4 Draw_request(flag)
136 cC3  BOOL flag;
137 GW  register struct RastPort *rp;
138 kX  register LONG newgidid,success;
139 Dk  register int i;
140 Fx  static BOOL execute;
141 Lv  static struct Gadget ok_cancel[2];
142 YP  if(execute == 0){
143 bO6      REQ_requester.ReqGadget=NULL;
144 D3      ok_cancel[0] = FRM_cancel;
145 Ar      ok_cancel[1] = FRM_ok;
146 iE      ok_cancel[0].NextGadget = 0;
147 B7      ok_cancel[0].LeftEdge=5;
148 8p      ok_cancel[0].TopEdge=35;
149 aa      ok_cancel[0].Activation = ENDGADGET;
150 yM      ok_cancel[0].GadgetType = REQGADGET;
151 ji      ok_cancel[0].GadgetID=OK;
152 Ct      ok_cancel[0].TopEdge=35;
153 rO      ok_cancel[1].NextGadget = 0;
154 iI      ok_cancel[1].LeftEdge=139;
155 iJ      ok_cancel[1].Activation = ENDGADGET;
156 6V      ok_cancel[1].GadgetType = REQGADGET;
157 3m      ok_cancel[1].GadgetID=CANCEL;
158 K2      ok_cancel[1].TopEdge=35;
159 Sg      AddGList(Main_Window,&REQ_name, 1L,1L,&REQ_request
er);
160 M8      AddGList(Main_Window,&ok_cancel[0],2L,1L,&REQ_request
er);
161 TH      AddGList(Main_Window,&ok_cancel[1],3L,1L,&REQ_request
er);
162 QJ      execute = 1;
163 JN3      success = Request(&REQ_requester,Main_Window);
164 j4      rp = REQ_requester.ReqLayer->rp;
165 ZA      DrawImage(rp,&FRM_Image8,-1L,-40L);
166 4Y      DrawImage(rp,&FRM_Image9,-168L,-40L);
167 fd      SetAPen(rp,1L);
168 Yp      drawbox(rp,1L,1L,196L,50L);
169 7H      for(i=0;i<=4;i++){
170 hY6          Move(rp,9L,(long)(7+i*2));
171 qR          Draw(rp,185L,(long)(7+i*2));
172 59          Move(rp,62L,(long)(38+i*2));
173 VO          Draw(rp,135L,(long)(38+i*2));
174 p63          if(flag == 0) DrawImage(rp,&REQ_Image2,52L,2L);
175 zL          if(flag == 1) DrawImage(rp,&REQ_Image1,57L,2L);
176 XB          if(flag == 2) DrawImage(rp,&command_img,45L,2L);
177 ZN          Delay(5L);
178 Zy          success=ActivateGadget(&REQ_name,Main_Window,&REQ_reques
ter);

```

Listing 4. »sc2.c« bitte mit dem Checksummer eingeben

```

179 Fe0 /*****
180 cJW *****/
181 LVO Mon May 02 21:04:45 1988
182 Nm *****/
183 JG Call_Rename(flag)
184 Oy3     BOOL flag;
185 Vu     struct Gadget *igad;
186 vA     struct IntuiMessage *Wait_for_Message();
187 k4     struct IntuiMessage *Message;
188 4k     struct Lock *ignorelock,*dirlock;
189 gy     register LONG number,gadid,newgadid;
190 Br     LONG Wait_for_Rename();
191 WY     char newdir[500];
192 qFO /*=====
193 3m4     mkdir
194 IA1     =====*/
195 pA3     if(flag == 1){
196 xD6         mkdir();
197 eZ         return(0);
198 wLO /*=====
199 NF1     =====*/
200 Ev3     WriteMessage("please select a File.");
201 fp1     FOREVER[
202 FF6         Message=Wait_for_Message(Main_Window,0);
203 oN         gadid = ((struct Gadget *) Message->IAddress)->Gad
etID;
204 Nw         if(Block_Control(gadid) == 1) continue;
205 fb         if( gadid >= 3 && gadid <= 15 ){
206 Sw9             WriteMessage("");
207 hm                 strcpy(r_m_buffer,src.list.name[gadid-3+src.scroll
_count]);
208 ba                 newgadid=Wait_for_Rename(Message,flag);
209 Ok                 if(newgadid == -1) return(-1);
210 B8                 RenameFile(src.list.name[gadid-3+src.scroll_count]
,r_m_buffer,SOURCE);
211 D1                 strcpy(src.list.name[gadid-3+src.scroll_count],r_m
_buffer);
212 Bm                 ClearBlock(SOURCE,(UWORD)gadid-3+src.scroll_count)
;
213 vp                 DisplayFiles(SOURCE);
214 J9                 return();
215 eT6         if( gadid >= 48 && gadid <= 61 ){
216 o69             WriteMessage("");
217 wD                 strcpy(r_m_buffer,dst.list.name[gadid-48+dst.scrol
l_count]);
218 lk                 newgadid=Wait_for_Rename(Message,flag);
219 Au                 if(newgadid == -1) return(-1);
220 lW                 RenameFile(dst.list.name[gadid-48+dst.scroll_count
],r_m_buffer,DESTINATION);
221 Ry                 strcpy(dst.list.name[gadid-48+dst.scroll_count],r_
m_buffer);
222 rQ                 ClearBlock(DESTINATION,(UWORD)gadid-48+dst.scroll
_count);
223 e4                 DisplayFiles(DESTINATION);
224 TJ                 return();
225 ls3     } /* end of forever */
226 sIO } /* end of block */
227 Po /*=====
228 P11 Baue Rename-mkdir Req auf
229 O1     0 = Rename
230 Hg     1 = MakeDir
231 t1     =====*/
232 f80 LONG Wait_for_Rename(Message,flag)
233 Uo3     struct IntuiMessage *Message;
234 Cm     BOOL flag;
235 ix     struct IntuiMessage *Wait_for_Message();
236 K7     register LONG newgadid,succes;
237 nK     register int i;
238 rF     Draw_request((BOOL)flag);
239 ex     Message=Wait_for_Message(Main_Window,(ULONG)0);
240 t2     if(Message == 0) return((long)-1);
241 a1     if( (newgadid = ((struct Gadget *) Message->IAddress)-
>GadgetID) == R_M){
242 pX6         EndRequest(&REQ_requester,Main_Window);
243 Ah         return((long)newgadid);
244 Nr3         if((newgadid==OK)){
245 sa6             EndRequest(&REQ_requester,Main_Window);
246 Dk             return((long)newgadid);
247 rF3         else return(-1);
248 k90 /*=====
249 PT4         Rename't ein File
250 C41     =====*/

```

```

251 kY0 RenameFile(oldname,newname,flag)
252 do3     char oldname[],newname[];
253 M0     LONG flag;
254 wI     char oldfilename[500];
255 oL     char newfilename[500];
256 EO     switch(flag){
257 aA     case SOURCE:
258 6PC         namecat(oldname,oldfilename,SOURCE);
259 OE         namecat(newname,newfilename,SOURCE);
260 7G         break;
261 sY3     case DESTINATION:
262 JYC         namecat(oldname,oldfilename,DESTINATION);
263 dN         namecat(newname,newfilename,DESTINATION);
264 BK         break;
265 pS3     } /* end of switch */
266 5XC         return(Rename(oldfilename,newfilename));
267 zt0 } /* end of Block */
268 g5 /*****
269 7yU *****/
270 pr0 Sat May 07 14:56:21 1988
271 oD *****/
272 lU Delete(){
273 HU3     register ULONG i,t=0;
274 86     struct FileLock *oldlock,*newlock;
275 sQ     char file[500];
276 fO     LONG block_type;
277 eO     long checkblocks();
278 ge     ULONG error;
279 UN     block_type = checkblocks();
280 CZ     switch(block_type){
281 yY6         case SOURCE:
282 csC             for(i=0;i<src.list.entrys;i++){
283 W5F                 if(src.list.select[i] == 1){
284 T8I                     if(src.list.dir[i] == 1){
285 yuE                         CopyDir(src.list.name[i],1L,0);
286 nLL                             namecat(src.list.name[i],file,block_ty
pe);
287 aM                             kill_file_entry(block_type,i);
288 HCI                     else{
289 nWL                         t++;
290 l7                         WriteMessage(src.list.name[i]);
291 sQ                             namecat(src.list.name[i],file,block_ty
pe);
292 ll                             if(DeleteFile(file) == NULL){
293 kXO                                 detect_errors();
294 nO                                 return(-1);
295 aNL                             else
296 jVO                                 kill_file_entry(block_type,i);
297 npI                             i--;
298 ka                                 src.scroll_count=0;
299 KwC                                 RefreshGList(&MAIN_s_path,Main_Window,NULL,1L);
300 lu                                 break;
301 WC6         case DESTINATION:
302 buC             for(i=0;i<dst.list.entrys;i++){
303 HoF                 if(dst.list.select[i] == 1){
304 ErI                     if(dst.list.dir[i] == 1){
305 jdE                         CopyDir(dst.list.name[i],1L,0);
306 Y4L                             namecat(dst.list.name[i],file,block_ty
pe);
307 ug                             kill_file_entry(block_type,i);
308 bWI                     else{
309 7GL                         t++;
310 Tq                             WriteMessage(dst.list.name[i]);
311 d9                             namecat(dst.list.name[i],file,block_ty
pe);
312 55                             if(DeleteFile(file) == NULL){
313 4rO                                 detect_errors();
314 71                                 return(-1);
315 uhL                             else
316 3pO                                 kill_file_entry(block_type,i);
317 79I                             i--;
318 4u                                 src.scroll_count=0;
319 CmC                                 RefreshGList(&MAIN_d_path,Main_Window,NULL,1L);
320 5E                                 break;
321 xf3         ClearAllSelected((ULONG)block_type,0);
322 8y         src.scroll_count=0;
323 oh         dst.scroll_count=0;
324 n7         PropGad(block_type,0);
325 uA         PropGad(block_type,1);
326 ll         DisplayFiles(block_type);
327 cR         Write_Info(block_type,2,0);
328 e30 /*****

```



```

*****
329 71N          BCPL-String in C-String umwandeln
330 IEO Fri Apr 29 20:49:57 1988
331 mB *****
*****
332 fo BstrC(bstr,buffer)
333 mr3  BSTR *bstr;
334 du  UBYTE *buffer;
335 dP  UBYTE *str;
336 qG  LONG loop,zaehler=0;
337 16  str = (UBYTE *) BADDR( bstr );
338 OP  for( loop = (LONG) str[0]; loop--; ++zaehler){
339 ZX6  buffer[zaehler] = str[zaehler+1];
340 4V3  buffer[zaehler] = 0;
341 rGO  /*****
*****
342 5aM          Registriere alle Devices im Speicher
343 sZ0 Fri Apr 29 20:57:12 1988
344 zO *****
*****
345 gB Record_Volumes(){
346 8H3  register ULONG xStart=304,yStart=97,count=0;
347 nx  char buffer[10];
348 Xa  struct RootNode *rootnode;
349 4Z  struct DosInfo *dosinfo;
350 dh  struct DeviceList *devicelist;
351 Kp  struct FileLock *filelock;
352 NU6  rootnode = (struct RootNode *) DOSBase->dl_Root;
353 Gp  dosinfo = (struct DosInfo *) BADDR( rootnode->rn_In
fo);
354 vt  devicelist = (struct DeviceList *) BADDR( dosinfo->d
l_DevInfo );
355 JP  SetAPen(mainRP,2L);
356 hn9  while( count <=29 ){
357 BzC  if( devicelist->dl_Type == 0){ /* nur wenn De
vice */
358 WBF  BstrC(devicelist->dl_Name,buffer);
359 tx  if( (checkdevice(buffer)) == 0) goto next;
360 O5  strcpy(vlm.n[count++],buffer);
361 1C  vlm.entrys++;
362 mH  if( count <= 4){
363 qZI  if( strlen(buffer) <= 6 ){
364 8BL  Move(mainRP,(long)xStart,(long)yStart+
(count-1)*10);
Text(mainRP,vlm.n[count-1],(long)strle
n(vlm.n[count-1]) );
365 kP  if(devicelist->dl_Next == 0) break;
366 ZkC  next:
367 kD0  devicelist = (struct DeviceList *) BADDR( devic
elist->dl_Next );
368 GuC  PropGad((ULONG)2,(UBYTE)1);
369 6k9  PropGad((ULONG)2,(UBYTE)0);
370 1J  /*****
371 j80  *****
372 Dv4  Checke Device
373 B31  *****
374 uN0  checkdevice(device)
375 9h3  char device[];
376 uv  static char *devlist[] = { "PRT", "SER", "CON", "RAW", "PAR"
};
377 O3  register LONG i,n;
378 p7  for( i=0; i<=4; i++){
379 tc6  if( (strcmp(device,devlist[i])) == NULL){
380 Yd9  return(OL);
381 dj3  return(1L);
382 WvO  /*****
*****
383 7gJ          Schreibt eine Message in die Message-Lin
e
384 jQ0 Mon May 02 21:06:22 1988
385 e3 *****
*****
386 LA WriteMessage(text)
387 Be3  char text[];
388 Lk  strcpy(MessageLine,text);
389 Bp  RefreshGLst(&MAIN_message,Main_Window,NULL,1L);
390 e30  /*****
*****
391 MzK          Warte auf Message von beliebigem Window
392 iy0 Wed May 04 20:37:43 1988
393 mB *****
*****
394 FE struct IntuiMessage *Wait_for_Message(MsgWindow,Code)
395 7N3  struct Window *MsgWindow;
396 Z2  ULONG Code;

```

```

397 p3  register ULONG i;
398 9T  struct IntuiMessage *Message;
399 8C2  static struct IntuiMessage Message2;
400 IQ3  BYTE *source,*destination;
401 112  FOREVER
402 r76  Wait((LONG)1<<MsgWindow->UserPort->mp_SigBit);
403 Z15  while ((Message = (struct IntuiMessage *) GetMsg ((str
uct MsgPort *)MsgWindow->UserPort)))
404 hF9  source = (BYTE *)Message; destination = (BYTE *)&M
essage2;
405 OX  for(i=0;i<sizeof(struct IntuiMessage);i++){
406 GM8  *destination++=*source++;
407 sI6  switch(Code){
408 aQC  case 0:
409 e1F  if( (Message->Class == CLOSEWINDOW)){return
(0);}
410 Z1  if( (Message->Class == REQCLEAR)){return(0)
;}
411 pJ  if( (Message->Class == GADGETUP) || (Messag
e->Class == GADGETDOWN)
412 JHL  || (Message->Class == MOUSEBUTTONS)){
413 rPH  ReplyMsg (Message);
414 ZTI  return((struct IntuiMessage *)&Message2);
415 c1  break;
416 k1C  case 1:
417 kpI  ReplyMsg(Message);
418 aV  return(&Message2);
419 gp  break;
420 8XO  /*****
*****
421 2OP          File an Pfad und in Zielbuffer
422 zC0 Sat May 07 14:59:47 1988
423 Gf *****
*****
424 I7  namecat(pfad,destination,flag)
425 JM3  char *pfad,*destination;
426 9n  LONG flag;
427 z9  switch(flag){
428 Lv6  case SOURCE:
429 T19  if((strlen(pfad)+strlen(src.spath)) >= 180){
430 HZC  WriteMessage("Path-Memory full !");
431 BM  MAIN_WaitMsg();
432 5R9  if(src.depth == 0)
433 QUC  sprintf(destination,"%s%s",src.spath,pfad);
434 pc9  else
435 hDC  sprintf(destination,"%s/%s",src.spath,pfad);
436 x69  break;
437 iO6  case DESTINATION:
438 oN9  if((strlen(pfad)+strlen(dst.dpath)) >= 180){
439 nGC  WriteMessage("Path-Memory full!");
440 KV  MAIN_WaitMsg();
441 fz9  if(dst.depth == 0)
442 b6C  sprintf(destination,"%s%s",dst.dpath,pfad);
443 y19  else
444 JYC  sprintf(destination,"%s/%s",dst.dpath,pfad);
445 6F9  break;
446 uo4  default:
447 bj9  if(destination[strlen(destination)-1] == ':'){
448 JoC  strcat(destination,pfad);
449 sn9  else{
450 Q1C  strcat(destination,"/");
451 mr  strcat(destination,pfad);
452 e30  /*****
*****
453 mJL          alle selektierten Files deselektieren
454 DRO Sat May 07 16:48:25 1988
455 mB *****
*****
456 5L  ClearAllSelected(flag,kill_dirs)
457 tr3  ULONG flag,kill_dirs;
458 o2  register ULONG i;
459 Vf2  switch(flag){
460 rR6  case SOURCE:
461 1X9  for(i=0;i<=src.list.entrys;i++){
462 smC  src.list.select[i]=0;
463 Mq  if(kill_dirs==1) src.list.dir[i]=0;
464 PY6  break;
465 Aq  case DESTINATION:
466 oN9  for(i=0;i<=dst.list.entrys;i++){
467 cZC  dst.list.select[i]=0;
468 CM  if(kill_dirs==1) dst.list.dir[i]=0;
469 Ud6  break;
470 J63  EraseBlock(flag);

```

Listing 4. (Fortsetzung)

```

471 b2 DisplayFiles(flag);
472 yNO /*****
*****
473 6eT alle Files selektieren
474 p10 Sat May 28 13:49:09 1988
475 6V /*****
*****/
476 bK SetAll(flag)
477 yc3 LONG flag;
478 8M register ULONG i;
479 pz2 switch(flag){
480 B16 case SOURCE:
481 Lr9 for(i=0;i<=src.list.entries;i++){
482 I8C src.list.select[i]=1;
483 ir6 break;
484 T9 case DESTINATION:
485 7g9 for(i=0;i<=dst.list.entries;i++){
486 luC dst.list.select[i]=1;
487 mv6 break;
488 b03 EraseBlock(flag);
489 tK DisplayFiles(flag);
490 Gf0 /*****
*****
491 I7M loeschen eines Fileeintrags im Block
492 160 Sat May 07 18:22:55 1988
493 On /*****
*****/
494 d1 kill_file_entry(flag,pos)
495 IL3 register LONG flag,pos;
496 HU register LONG i;
497 7H switch(flag){
498 T36 case SOURCE:
499 wM9 for(i=pos;i<src.list.entries;i++){
500 LkC strcpy(src.list.name[i],src.list.name[i+1]);
501 EA src.list.select[i]=src.list.select[i+1];
502 XW src.list.dir[i] =src.list.dir[i+1];
503 kH src.list.length[i]=src.list.length[i+1];
504 3d9 src.list.entries--;
505 I9 EraseBlock(SOURCE);
506 5E break;
507 qW6 case DESTINATION:
508 qW9 for(i=pos;i<dst.list.entries;i++){
509 ezC strcpy(dst.list.name[i],dst.list.name[i+1]);
510 Rg dst.list.select[i]=dst.list.select[i+1];
511 k2 dst.list.dir[i] =dst.list.dir[i+1];
512 xn dst.list.length[i]=dst.list.length[i+1];
513 rU9 dst.list.entries--;
514 5r EraseBlock(DESTINATION);
515 EN break;
516 g50 /*****
*****
517 EAI hineinsetzen eines Filenames in einem Blo
ck
518 w20 Sat May 07 18:40:32 1988
519 oD /*****
*****/
520 NV insert_file_entry(old_position)
521 E73 ULONG old_position;
522 PV register int i=0;
523 oe char str1[180],str2[180],*lcase();
524 by if(dst.list.entries > 0){
525 Ok6 while(i <= dst.list.entries){
526 IS9 lcase(str1,dst.list.name[i]);
527 mt lcase(str2,src.list.name[old_position]);
528 im if(strcmp(str1,str2) == 0)
529 aBC return(-1);
530 KI7 i++;
531 PB3 strcpy(dst.list.name [dst.list.entries],src.list.name [
old_position]);
532 No dst.list.dir [dst.list.entries]=src.list.dir [old_pos
ition];
533 yb dst.list.length[dst.list.entries]=src.list.length[old_pos
ition];
534 Nq Sort_Files(DESTINATION);
535 3c dst.list.entries++;
536 d10 mkdir(){}
537 Ba2 struct Gadget *igad;
538 bq3 struct IntuiMessage *Wait_for_Message();
539 Qk2 struct IntuiMessage *Message;
540 kQ3 struct Lock *ignorelock,*dirlock;
541 Me register LONG number,gadid,newgadid;
542 rX LONG Wait_for_Rename();
543 CE char newdir[500];
544 z1 LONG block_type;

```

```

545 qb if( (newgadid=Wait_for_Rename(Message,(BOOL)1)) == -1){
546 r86 return(-1);
547 tN3 block_type=checkblocks();
548 Wt switch(block_type){
549 Is6 case SOURCE:
550 51C namecat(r_m_buffer,newdir,SOURCE);
551 aV strcpy(src.list.name[src.list.entries],r_m_buffe
r);
552 QV src.list.dir[src.list.entries]=1;
553 hD src.list.select[src.list.entries]=0;
554 hD src.list.entries++;
555 s18 break;
556 dJ4 case DESTINATION:
557 W6C namecat(r_m_buffer,newdir,DESTINATION);
558 Ao strcpy(dst.list.name[dst.list.entries],r_m_buffe
r);
559 dJ dst.list.dir[dst.list.entries]=1;
560 Bn dst.list.select[dst.list.entries]=0;
561 T2 dst.list.entries++;
562 z88 break;
563 X23 if( (ignorelock = Lock(newdir,ACCESS_READ)) == NULL){
564 FK6 WriteMessage("creating a new directory.");
565 8U dirlock=CreateDir(newdir);
566 IS UnLock(dirlock);
567 Ka3 if(ignorelock != NULL) UnLock(ignorelock);
568 Ts Sort_Files(block_type);
569 SS EraseBlock(block_type);
570 L5 PropGad(block_type,0);
571 s8 PropGad(block_type,1);
572 zz DisplayFiles(block_type);
573 b00 /*****
*****
574 Uya Command
575 Hb0 Sat May 28 17:27:16 1988
576 J8 /*****
*****/
577 o5 command()
578 qF2 struct Gadget *igad;
579 GV3 struct IntuiMessage *Wait_for_Message();
580 oI char *strchr();
581 6Q2 struct IntuiMessage *Message;
582 Q63 struct Lock *ignorelock,*dirlock;
583 2K register LONG number,gadid,newgadid;
584 gN struct FileHandle *fp;
585 YE LONG Wait_for_Rename();
586 oJ2 static char *argv[20];
587 264 int i=0,t=0,n=0;
588 dK3 if( (newgadid=Wait_for_Rename(Message,(BOOL)2)) == -1){
589 Y96 return(-1);
590 Tg3 Execute(r_m_buffer,0L,0L);
591 88 WindowToFront(Main_Window);
592 uJ0 /*****
*****
593 Hwa SHOWPIC
594 VNO Fri Jun 03 19:10:19 1988
595 2R /*****
*****/
596 Hx showpic(){
597 xM2 static char *argv[50],*execname;
598 MI3 register int i,n=1;
599 cK LONG checkblocks(),totalsize=0,bl_type;
600 VJ static UBYTE callname[] = "c:show ";
601 YB struct FileLock *lock;
602 nV2 struct FileLock *altlock;
603 AM3 argv[0] = callname;
604 Pe bl_type = checkblocks();
605 Mu switch(bl_type){
606 Dn6 case SOURCE:
607 tJ9 lock = Lock(src.spath,ACCESS_READ);
608 vO altlock=CurrentDir(lock);
609 kt break;
610 VB4 case DESTINATION:
611 zq9 lock = Lock(dst.dpath,ACCESS_READ);
612 zS altlock=CurrentDir(lock);
613 ox break;
614 V33 switch(bl_type){
615 Mw6 case SOURCE:
616 OG9 for(i=0;i<src.list.entries;i++){
617 uTC if(src.list.select[i] == 1){
618 qLB argv[n++] = src.list.name[i];
619 u39 break;
620 fL6 case DESTINATION:
621 k39 for(i=0;i<dst.list.entries;i++){
622 QxC if(dst.list.select[i] == 1){

```



```

623 gHB      argv[n++] = dst.list.name[i];
624 z89      break;
625 R53      for(i=1;i<n;i++){
626 yC6      totalsize+=strlen(argv[i]);
627 yM3      totalsize+=n+6; /* fuer Show-Befehl und Spaces */
628 oL      execname = (char *) AllocMem(totalsize, MEMF_CLEAR);
629 4z      strcpy(execname, callname);
630 WA      for(i=1;i<n;i++){
631 p96      strcat(execname, argv[i]);
632 de      if(i < (n-1)) strcat(execname, " ");
633 Xs3      Execute(execname, 0L, 0L);
634 vU      for(i=0; i<=49; i++){
635 cr6      argv[i]=0;
636 QK3      FreeMem(execname, totalsize);
637 ch      Unlock(lock);
638 FH      CurrentDir(allock);
639 PS      ClearAllSelected(bl_type, 0);
640 H6      DisplayFiles(bl_type);
641 h60      /*****
*****
642 1LX      ROOT-Directory
643 Uw0      Sat May 28 19:19:07 1988
644 pE      *****/
*****/
645 J5      root_dir(){
646 OS3      char newdir[80], *strchr();
647 zI      switch(checkblocks()){
648 tT9      case SOURCE:
649 JmC      strcpy(newdir, src.spath);
650 gS3      newdir[strchr(newdir, ':')-newdir+1]=0;
651 f6      strcpy(src.spath, newdir);
652 fK      src.depth = 0;
653 Sb      break;
654 Dt9      case DESTINATION:
655 b5C      strcpy(newdir, dst.dpath);
656 mo      newdir[strchr(newdir, ':')-newdir+1]=0;
657 nf      strcpy(dst.dpath, newdir);
658 Q8      dst.depth = 0;
659 Yn      break;
660 PS3      ClearAll(checkblocks());
661 iI      RefreshGList(&MAIN_d_path, Main_Window, NULL, 1L);
662 Bn      RefreshGList(&MAIN_s_path, Main_Window, NULL, 1L);
663 Mo      ReadDirectory(newdir, checkblocks());
664 4T0      /*****
*****
665 VJY      SWAP-Blocks
666 uN0      Sat May 28 19:19:26 1988

```

```

667 Cb      *****/
*****/
668 mx      swap(){
669 2G3      struct Source source;
670 eh      source = src;
671 dY      src = dst;
672 Qf      dst = source;
673 mV      print_files(SOURCE);
674 15      print_files(DESTINATION);
675 ov      MAIN_MAIN_s_pathSInfo.Buffer = src.spath;
676 ZF      MAIN_MAIN_d_pathSInfo.Buffer = dst.dpath;
677 yY      RefreshGList(&MAIN_d_path, Main_Window, NULL, 1L);
678 R3      RefreshGList(&MAIN_s_path, Main_Window, NULL, 1L);
679 2r0      detect_errors(){
680 RS3      LONG error;
681 jY      error = IoErr();
682 Ob      switch(error){
683 rh6      case 202:
684 SG9      WriteMessage("Object is in use !");
685 y7      break;
686 1t6      case 213:
687 9L9      WriteMessage("Disk is not validated.");
688 1A      break;
689 zq6      case 221:
690 Em9      WriteMessage("Disk is full.");
691 4D      break;
692 6y6      case 222:
693 e49      WriteMessage("File is protected from deletion.");
694 7G      break;
695 D66      case 223:
696 369      WriteMessage("File is protected from writing.");
697 AJ      break;
698 OJ6      case 225:
699 MH9      WriteMessage("Not a DOS disk.");
700 DM      break;
701 VR6      case 226:
702 BS9      WriteMessage("No disk in drive.");
703 GP      break;
704 E46      case 103:
705 wy9      WriteMessage("Insufficient free store.");
706 JS      break;
707 714      default:
708 OD9      WriteMessage("Disk-Error.");
(C) 1988 M&T

```

Listing 4. (Schluß)

Programmname: sc3.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Bemerkung: siehe Listing 1

```

1 Nm0 /*****
*****
2 Sc6      Supercopy v1.0 (w) in 1988 by Eichenberger Software P
      roduction
3 HbR      Window-Aktionen Funktionen
4 3G0      Tue Apr 26 20:20:41 1988
5 3w      *****/
*****SC3*/
6 qF      /*****
7 tU4      HEADERS
8 IA1      *****/
9 Vy0      #include<functions.h>
10 ds      #include<:sc_struct.h>
11 vK      /*****
12 1v4      DEFINES
13 NF1      *****/
14 Jw0      #define SOURCE 0L
15 v5      #define DESTINATION 1L
16 JK      #define UP 100
17 bL      #define DOWN 101
18 2R      /*****
19 ZB4      GLOBALE VARIABLEN
20 UM1      *****/
21 JL0      struct Source src;
22 Hu      struct Destination dst;
23 st      struct Volumes vlm;
24 7J      LONG Wait_on_Message();
25 GR      extern struct GfxBase *GfxBase;

```

```

26 sk      extern struct RastPort *mainRP;
27 V4      extern struct Window *Main_Window;
28 gZ      extern struct Gadget MAIN_source, MAIN_destination, MAIN_s_pa
      th, MAIN_d_path,
29 B1L      MAIN_s_prop, MAIN_d_prop, MAIN_dev_prop;
30 Ou0      extern struct PropInfo MAIN_MAIN_s_propSInfo, MAIN_MAIN_d_pr
      opSInfo, MAIN_MAIN_dev_propSInfo;
31 bg      USHORT oldcolor[5];
32 sH      /*****
*****
33 EWV      Warte auf Message
34 vt0      Sun May 01 16:16:23 1988
35 OP      *****/
*****/
36 O3      MAIN_WaitMsg(){
37 xp2      register ULONG Class;
38 r2      register USHORT Code, Qualifier;
39 Gt3      register SHORT mouseX, mouseY;
40 NO2      register struct IntuiMessage *Message;
41 DY3      register struct Window *IDCMPwindow;
42 Ag2      int i;
43 Fw      FOREVER
44 O86      Wait((LONG)1<<Main_Window->UserPort->mp_SigBit);
45 x95      while ((Message = (struct IntuiMessage *) GetMsg ((str
      uct MsgPort *)Main_Window->UserPort)))
46 Sq7      Class = Message->Class;
47 HB8      Code = Message->Code;
48 tD9      mouseX= Message->mouseX;
49 OM      mouseY= Message->mouseY;
50 O7      Qualifier= Message->Qualifier;
51 gr6      IDCMPwindow = Message->IDCMPwindow;
52 2a8      ReplyMsg (Message);
53 jd6      switch (Class){

```

Listing 5. »sc3.c« bitte mit dem Checksummer eingeben

```

54 OSC      case RAWKEY:
55 JW        Key_Call(Code,Qualifier);
56 pyI        break;
57 LfC      case ACTIVEWINDOW:
58 L2I        MoveWindow(Main_Window,OL,(long)-(Main_Wi
59 wO        ndow->TopEdge));
60 t2        SizeWindow(Main_Window,OL,(long)256-(Main
61 rF8        _Window->Height));
62 BQI        break;
63 Zg      case MOUSEBUTTONS :
64 x6T        if(Code != MENUDOWN) break;
65 2t9        Call_Mouse(MouseX,MouseY,Code);
66 z8S        break;
67 Z09      case REQCLEAR :
68 1AS        break;
69 bPA      case GADGETDOWN : do_gadgets(Message);
70 3CS        break;
71 fm8      case GADGETUP : do_gadgets(Message);
72 Sj        break;
73 Wq4      } /* End of switch */
74 Yt2      Class = 0; Code = 0;
75 m90      } /* End of while */
76 az      } /* End of forever */
77 hvT      } /* End of Wait_Msg */
78 HNO      /* *****
79 17      *****
80 ks1      do_gadgets(mes) /* Frage welches Gadget gedruickt wurde,
81 voQ      danach die
82 1j1      gewaehlten Commands ausfuehren ..
83 rG3      */
84 5E      struct IntuiMessage *mes;
85 VS      struct Gadget *igad;
86 9r      BOOL Block_Control(),success;
87 ax      register LONG gadid,1,number,number2,number3;
88 li      igad = (struct Gadget *) mes->IAddress; /* Ptr auf Gad
89 ns      get */
90 7B      gadid = igad->GadgetID; /* Eigene Identitaetsnummer */
91 Dc0      number = (gadid-3)+src.scroll_count;
92 pa4      number2 = (gadid-19)+v1m.scroll_count;
93 fX1      number3 = (gadid-48)+dst.scroll_count;
94 N23      /* *****
95 4K8      SOURCE-Block
96 FW7      *****
97 OB5      if( (gadid >= 3 && gadid <= 15) && (number <= src.lis
98 EUA      t.entries-1) ){
99 kH7      if(src.list.select[ number ] == 0)
100 60      src.list.select[ number ]=1;
101 Nm0      else
102 D44      src.list.select[ number ]=0;
103 ph1      ClearBlock(SOURCE,gadid-3);
104 7S3      DisplayFiles(SOURCE);
105 Z18      /* *****
106 q07      DESTINATION-Block
107 YL5      *****
108 oLA      if( (gadid >= 48 && gadid <= 61) && (number3 <= dst.1
109 gR7      st.entries-1) ){
110 pF      if(dst.list.select[ number3 ] == 0)
111 Xw0      dst.list.select[ number3 ]=1;
112 We4      else
113 zr1      dst.list.select[ number3 ]=0;
114 5y3      ClearBlock(DESTINATION,gadid-48);
115 1K9      DisplayFiles(DESTINATION);
116 JtC      /* *****
117 WqI      VOLUMES-Block
118 UN      *****
119 VP      if( (gadid >= 19 && gadid <= 27) && (number2 <= v1m.e
120 8b      ntrys-1) ){
121 S4      switch(checkblocks() ){
122 wb      case SOURCE:
123 tU      src.depth=0;
124 v4      src.list.entries=0;
125 gMC      strcpy(src.spath,v1m.n[ number2 ]);
126 KhI      strcat(src.spath,":");
RefreshGList(&MAIN_s_path,Main_Window,NUL
L,IL);
ClearAll(SOURCE);
Read_Directory(src.spath,SOURCE);
break;
case DESTINATION:
dst.depth=0;

```

```

127 IE      dst.list.entries=0;
128 g1      strcpy(dst.dpath,v1m.n[ number2 ]);
129 JD      strcat(dst.dpath,":");
130 9J      RefreshGList(&MAIN_d_path,Main_Window,NUL
131 Ay      L,IL);
132 TB      ClearAll(DESTINATION);
133 4D      Read_Directory(dst.dpath,DESTINATION);
134 ID6      break;
135 el3      } /* end of switch */
136 GD      } /* end of if */
137 Im      success=Block_Control(gadid);
138 lh      if(success == 1) return();
139 509      switch(gadid){
140 vYF      case COPY:
141 CL      Copy();
142 MJ9      break;
143 4wF      case RENAME:
144 FO      Call_Rename((BOOL)0);
145 pY6      break;
146 pGF      case DELETE:
147 IR      Delete();
148 619      break;
149 51F      case FORMAT:
150 LU      InitFormatReq();
151 hf9      break;
152 Z6A      case COMMAND:
153 OXF      command();
154 gI9      break;
155 pFF      case BYTES:
156 Ra      call_bytes();
157 TK9      break;
158 TsF      case MOVE:
159 Ud      Copy((BOOL)1);
160 WD9      break;
161 RKF      case MAKEDIR:
162 Xg      Call_Rename((BOOL)1);
163 tv9      break;
164 JTF      case TYPE:
165 aj      Type();
166 gT9      break;
167 oHF      case HTYPE:
168 dm      HType();
169 mh9      break;
170 foF      case SWAP:
171 tq9      break;
172 5nF      case DISKCOPY:
173 Ir      fexec1("c:diskcopy","c:diskcopy");
174 Qf9      break;
175 I6F      case SHOWPIC:
176 lu      showpic();
177 XP3      break;
178 bt0      } /* end of switch */
179 Fe      } /* end of block */
180 xHA      /* *****
181 yH0      untersucht, welcher Block aktiv ist SOURCE oder D
182 Nm      ESTINATION
183 Gk      Sat Apr 30 21:10:58 1988
184 YJ2      *****
185 I6      checkblocks(){
186 M10      if( MAIN_source.Flags & SELECTED) return(SOURCE);
187 QZ5      if( MAIN_destination.Flags & SELECTED) return(DESTINATION
188 Ph0      );
189 Ut      /* *****
190 08      *****
191 Fg3      Call_Mouse(x,y,Code)
192 ub      SHORT x,y; /* Mauskoordinaten */
193 5XC      USHORT Code;
194 VMF      if( (x >= 37) && (x <= 242) && (y <= 179) ){
195 HWI      if(src.list.dir[((y-52)/10) + src.scroll_cou
196 8n      nt] == 1){
197 5g      PathCat(src.list.name[((y-52)/10) + src.s
198 nrC      croll_count],SOURCE);
199 gYF      ClearAll(SOURCE);
Read_Directory(src.spath,SOURCE);
if( (x >= 401) && (x <= 506) && (y <= 179) )
{
if(dst.list.dir[((y-52)/10) + dst.scroll_cou

```



```

nt] == 1){
200 azI      PathCat(dst.list.name[((y-52)/10) + dst.s
202 bJI      croll_count],DESTINATION);
201 I6E      ClearAll(DESTINATION);
202 bJI      ReadDirectory(dst.dpath,DESTINATION);
203 Vv0 } /* end of block */
204 e3 /*****
*****
205 RvJ      Errechnet bei: what = 0 -> neue Positi
on
206 pwZ      what = 1 -> neue Groess
e
207 ZVO Sun May 01 16:15:40 1988
208 nC *****
*****
209 9U      PropGad(flag,what)
210 Ip3      ULONG flag;
211 J1      UBYTE what;
212 2y      switch (flag) {
213 sS6      case SOURCE:
214 kK      switch(what){
215 TjC      case 0:
216 xRF      if(src.list.entrys >= 14){
217 FHC      MAIN_MAIN_s_propSInfo.VertPot=((ULONG)src.scr
oll_count*65535/((src.list.entrys-1)-12));
218 L8F      else
219 BXC      MAIN_MAIN_s_propSInfo.VertPot=0;
220 TcF      break;
221 bsC      case 1:
222 3XF      if(src.list.entrys >= 14){
223 H1I      MAIN_MAIN_s_propSInfo.VertBody = (ULONG)6
5535/((src.list.entrys/13+1);
224 REF      else
225 dpI      MAIN_MAIN_s_propSInfo.VertBody = 65535;
226 Z18      break;
227 dv9      RefreshGList(&MAIN_s_prop,Main_Window,NULL,1L);
228 bk      break;
229 M26      case DESTINATION:
230 0a      switch(what){
231 JzC      case 0:
232 e6F      if(dst.list.entrys >= 14){
233 teD      MAIN_MAIN_d_propSInfo.VertPot=((ULONG)dst.scr
oll_count*65535/((dst.list.entrys-1)-12));
234 b0F      else
235 x4C      MAIN_MAIN_d_propSInfo.VertPot=0;
236 JsF      break;
237 r8C      case 1:
238 kCF      if(dst.list.entrys >= 14){
239 mfI      MAIN_MAIN_d_propSInfo.VertBody = (ULONG)6
5535/((dst.list.entrys/13+1));
240 hUF      else
241 RMI      MAIN_MAIN_d_propSInfo.VertBody = (ULONG)6
5535;
242 py8      break;
243 Rh9      RefreshGList(&MAIN_d_prop,Main_Window,NULL,1L);
244 ro      break;
245 nx6      case 2: /* ProportionalGadget fuer Devices und Volum
es */
246 Gq      switch(what){
247 zFC      case 0:
248 oGI      if(vlm.entrys >= 5)
249 6NC      MAIN_MAIN_dev_propSInfo.VertPot=((ULONG)vlm.scr
oll_count*65535/((vlm.entrys-1)-3));
250 reI      else
251 TZC      MAIN_MAIN_dev_propSInfo.VertPot=0;
252 z8F      break;
253 70C      case 1:
254 iIF      if(vlm.entrys >= 5){
255 MrI      MAIN_MAIN_dev_propSInfo.VertBody = 65535/
(vlm.entrys/4+1);
256 xkF      else
257 TFI      MAIN_MAIN_dev_propSInfo.VertBody = 65535;
258 5E8      break;
259 lt9      RefreshGList(&MAIN_dev_prop,Main_Window,NULL,1L);
260 7G      break;
261 l03 } /* end of switch */
262 az0 /*****
*****
263 eIR      Auswertung von RAWKEYCodes
264 dC0 Mon May 02 18:48:31 1988
265 17 *****
*****
266 63 Key_Call(Code,Qualifier)
267 i93 USHORT Code,Qualifier;

```

```

268 RX      register struct ViewPort *vp;
269 Z1      static BOOL resizing;
270 tY      vp = &(Main_Window->WScreen->ViewPort);
271 g62      switch(Code){
272 YJ6      case 0x50:
273 q69      SetRGB4(vp,0L,0x5L,0x5L,0x5L);
274 YN      SetRGB4(vp,1L,0xdL,0xbL,0x9L);
275 FI      SetRGB4(vp,2L,0x0L,0x0L,0x0L);
276 GJ      SetRGB4(vp,3L,0xfL,0x7L,0x0L);
277 OX      break;
278 Jv6      case 0x51:
279 Ki9      SetRGB4(vp,0L,0xaL,0xaL,0xaL);
280 r5      SetRGB4(vp,1L,0x2L,0x4L,0x6L);
281 Bq      SetRGB4(vp,2L,0xfL,0xfL,0xfL);
282 PT      SetRGB4(vp,3L,0x0L,0x8L,0xfL);
283 Ud      break;
284 u76      case 0x52:
285 vo9      SetRGB4(vp,0L,0x0L,0x5L,0xaL);
286 Dr      SetRGB4(vp,1L,0xfL,0xfL,0xfL);
287 zY      SetRGB4(vp,2L,0xcL,0xcL,0xfL);
288 VZ      SetRGB4(vp,3L,0x0L,0x8L,0xfL);
289 aj      break;
290 5J6      case 0x53:
291 Ph9      Recall_default_color();
292 dm      break;
293 DS6      case 0x54:
294 4R9      if(resizing == 0){
295 aOC      SizeWindow(Main_Window,(long)0,(long)-47);
296 Ms      MoveWindow(Main_Window,(long)0,(long)20);
297 Ip      resizing=1;
298 Js9      break;
299 Oe6      case 0x55:
300 Ca9      if(resizing == 1){
301 GxC      MoveWindow(Main_Window,0L,(long)-(Main_Window-
>TopEdge));
302 rv      SizeWindow(Main_Window,0L,(long)256-(Main_Windo
w->Height));
303 It      resizing=0;
304 py9      break;
305 Zq6      case 0x56:
306 mc      write_turbo_script();
307 s19      break;
308 KJ0 /*****
*****
309 upL      Sichert die aktuellen Workbench-Farben
310 8c0 Mon May 02 19:45:10 1988
311 Sr *****
*****
312 Qc      SaveDefaultColors(){
313 qv3      register BYTE i;
314 PY      for(i=0;i<=3;i++){
315 PE6      oldcolor[i] = GetRGB4(Main_Window->WScreen->ViewPor
t.ColorMap,(long)i);
316 Sr0 /*****
*****
317 OIC      Schreibt die Default-Farben wieder in die Color
Map ein.
318 zWO Mon May 02 20:21:08 1988
319 az *****
*****
320 OK      Recall_default_color(){
321 y33      register BYTE i;
322 md      register UWORD *ColorTable;
323 as      LoadRGB4(&(Main_Window->WScreen->ViewPort),oldcolor,4L
);
324 az0 /*****
*****
325 FRD      Rechnet Anzahl Bytes, die kopiert werden solle
n, aus.
326 RCO Sun Apr 24 17:13:28 1988
327 I7 *****
*****
328 kc      call_bytes(){
329 Ho3      register int i;
330 5z      register ULONG total=0;
331 RR      char text[10];
332 wD      switch(checkblocks()){
333 o06      case SOURCE:
334 S19      for(i=0;i<src.list.entrys;i++){
335 MvC      if(src.list.select[i] == 1){
336 PIF      total+=src.list.length[i];
337 MV9      break;
338 7n6      case DESTINATION:

```

Listing 5. (Fortsetzung)

```

339 CV9      for(i=0;i<dst.list.entries;i++){
340 sPC          if(dst.list.select[i] == 1){
341 5oF              total+=dst.list.length[i];
342 Ra9          break;
343 q13      sprintf(text,"%ld",total);
344 q62      WriteMessage(text);
345 vKO      /*****
346 sY3          totales CleanUp fuer Debugger
347 2R0      *****/
348 Na      _abort(){
349 eE3          puts("you have QUIT me !!!!");
350 aU      MEM_cleanup();
351 iW      MAIN_cleanup();
352 fAO      exit(TRUE);
353 3S      /*****
354 8pS          Fuer die Block-Kontrolle
355 QDO      Sun May 22 13:20:09 1988
356 Ba      *****/
357 Lr      BOOL Block_Controller(gadid)
358 k63      LONG gadid;
359 lI      register int i;
360 r7      register struct RastPort *rp;
361 NW      register ULONG xStart=304,yStart=97,count=0;
362 QJ      BOOL success;
363 2p      rp = mainRP;
364 PL      switch(gadid){
365 Ku9          case SOURCE:
366 gSF              if( (MAIN_source.Flags & SELECTED) == 0){
367 JCI                  MAIN_destination.Flags = GADGHIMAGE+GADGI
MAGE+SELECTED;
RefreshGList(&MAIN_destination,Main_Windo
w,NULL,1L);
368 2K          else{
369 aVF              MAIN_destination.Flags = GADGHIMAGE+GADGI
MAGE;
370 LuI              RefreshGList(&MAIN_destination,Main_Windo
w,NULL,1L);
371 5N          success=1;
372 kqF          break;
373 w5          case DESTINATION:
374 hn9              if( (MAIN_destination.Flags & SELECTED) == 0
){
375 NFF                  MAIN_source.Flags = GADGHIMAGE+GADGIMAG
E+
SELECTED;
RefreshGList(&MAIN_source,Main_Window,NUL
L,1L);
376 uhI          else{
377 mv              MAIN_source.Flags = GADGHIMAGE+GADGIMAG
E;
RefreshGList(&MAIN_source,Main_Window,NUL
L,1L);
378 jeF          success=1;
379 wPI          break;
380 py          case S_PATH:
381 tzF              ClearAll(SOURCE);
382 5EA              src.depth = NewPath(src.spath);
383 Rs6              if( (strcmp(src.spath,"ram:",4)) == NULL){
384 ApF                  strcpy(src.spath,"RAM:");
385 cyA                  Read_Directory(src.spath,SOURCE);
386 13F              success=1;
387 ZXI              break;
388 AlF          case D_PATH:
389 17              ClearAll(DESTINATION);
390 DM              dst.depth = NewPath(dst.dpath);
391 5H6              if( (strcmp(dst.dpath,"ram:",4)) == NULL){
392 NBF                  strcpy(dst.dpath,"RAM:");
393 c2A                  Read_Directory(dst.dpath,DESTINATION);
394 cNF              success=1;
395 j8I              break;
396 jRF          case SPROP:
397 9F              src.scroll_count = ((ULONG)(src.list.entries-
13))*
(ULONG)MAIN_MAIN_s_propSInfo.VertPot)/Oxf
fff;
398 LU          SetAPen(mainRP,0L);
399 8C9          RectFill(mainRP,32L,50L,250L,181L);
400 rmF          DisplayFiles(SOURCE);
success=1;
break;
case DPROP:
dst.scroll_count = ((ULONG)(dst.list.entries-
13))*
(ULONG)MAIN_MAIN_d_propSInfo.VertPot)/Oxf
fff;
401 z3A
402 RZF
403 zt
404 GM
405 SbA
406 la9
407 aHF

```

```

408 6AA      SetAPen(mainRP,0L);
409 QqF      RectFill(mainRP,396L,50L,614L,181L);
410 f5      DisplayFiles(DESTINATION);
411 NT      success=1;
412 Z1A      break;
413 Yg9      case DEVPROP:
414 geC          SetBPen(rp,0L);
415 lh          SetAPen(rp,2L);
416 5K          SetDrMd(rp,JAM2);
417 9tF          vlm.scroll_count = ((ULONG)(vlm.entries-4)*(U
LONG)MAIN_MAIN_dev_propSInfo.VertPot)/Oxffff;
418 BM          for(i=1;i<=4;i++){
419 6eI              if( strlen() <= 6 ){
420 15L                  Move(mainRP,(long)xStart,(long)yStart+
(1-1)*10);
421 No                  Text(mainRP,vlm.n[i-1+vlm.scroll_count
],(long)strlen(vlm.n[i-1+vlm.scroll_cou
nt]));
422 jsF          break;
423 Yb7      case QUIT:
424 mgF          MEM_cleanup();
425 u8          MAIN_cleanup();
426 rM          exit(TRUE);
427 dJ          success=1;
428 py          break;
429 8J9      case CLEAR:
430 lHF          ClearAllSelected((ULONG)checkblocks());
431 s1A          break;
432 KH9      case ALL:
433 JRF          SetAll((LONG)checkblocks());
434 v4          break;
435 e66      case SUP:
436 v1F          do{
437 AmE              if(src.scroll_count > 0){
438 CxL                  ScrollRaster(mainRP,0L,-10L,37L,52L,24
9L,179L);
439 3aF                  src.scroll_count--;
440 B1G                  DisplaySingle(SOURCE,100L);
441 kqL                  PropGad(SOURCE,0);
442 hSI                  Delay(2L);
443 6X9              }while(!(64 & *((UBYTE *)0xbfe001)));
444 u0F          success=1;
445 6F          break;
446 Ze9      case SDOWN:
447 6wF          do{
448 NdI              if((src.list.entries >= 14) && (src.scrol
l_count <= (src.list.entries-14))){
449 1wL                  ScrollRaster(mainRP,0L,10L,37L,52L,249
L,179L);
450 OTF                  src.scroll_count++;
451 R2G                  DisplaySingle(SOURCE,101L);
452 v1L                  PropGad(SOURCE,0);
453 sdI                  Delay(2L);
454 H1A              }while(!(64 & *((UBYTE *)0xbfe001)));
455 5BF          success=1;
456 HQ          break;
457 LL6      case DSTUP:
458 H7F          do{
459 xXE              if(dst.scroll_count > 0){
460 z3L                  ScrollRaster(mainRP,0L,-10L,401L,52L,6
13L,179L);
461 4eF                  dst.scroll_count--;
462 cLG                  DisplaySingle(DESTINATION,100L);
463 fHL                  PropGad(DESTINATION,0);
464 3oI                  Delay(2L);
465 St9              }while(!(64 & *((UBYTE *)0xbfe001)));
466 GMF          success=1;
467 Sb          break;
468 tJ9      case DSTDOWN:
469 SIF          do{
470 bqI              if((dst.list.entries >= 14) && (dst.scrol
l_count <= (dst.list.entries-14))){
471 WjL                  ScrollRaster(mainRP,0L,10L,401L,52L,61
3L,179L);
472 1XF                  dst.scroll_count++;
473 scG                  DisplaySingle(DESTINATION,101L);
474 qSL                  PropGad(DESTINATION,0);
475 EzI                  Delay(2L);
476 d4A              }while(!(64 & *((UBYTE *)0xbfe001)));
477 RXF          success=1;
478 dm          break;
479 hH9      case PARENT:
480 yMA          switch( checkblocks() ){
481 CmI              case SOURCE:
482 GpO                  Parent_Dir(src.spath,SOURCE);

```



```

483 IuG RefreshGList(&MAIN_s_path,Main_Window,NULL,
1L);
484 mR ClearAll(SOURCE);
485 jKO Read_Directory(src.spath,SOURCE);
486 lu break;
487 WCC case DESTINATION:
488 uBO Parent_Dir(dst.dpath,DESTINATION);
489 wWG RefreshGList(&MAIN_d_path,Main_Window,NULL,
1L);
490 xL ClearAll(DESTINATION);
491 GyO Read_Directory(dst.dpath,DESTINATIO
N);
492 rO break;
493 hnF success=1;
494 t2 break;
495 qu6 case ROOT:
496 qa root_dir();
497 w5F break;
498 509 case SWAP:
499 x6F swap();
500 z8 break;
501 FH9 default: success=0;break;
502 993 */end of switch*/
503 CZ2 return(success);
504 o10 */* end of Block */
505 Vu /*****
*****
506 cPI Testet ob Message vorliegt, wenn ja -> T
RUE
507 460 Wed Jun 01 11:07:46 1988
508 d2 *****/
509 tm LONG Wait_on_Message(window)
510 V13 struct Window *window;
511 OX register struct IntuiMessage *message;
512 Ea LONG gadid;
513 pV message = (struct IntuiMessage *)window->UserPort->mp_
MsgList.lh_Head;
514 uS while (message->ExecMessage.mn_Node.ln_Succ)
515 vQ gadid = (struct Gadget *) (message->IAddress->GadgetI
D;
516 Cz6 if (message->Class == GADGETUP) {
517 Ea9 message=GetMsg(window->UserPort);
518 TU ReplyMsg(message);
519 HC return(gadid);
520 Y16 if (message->Class == GADGETDOWN){
521 Ie9 message=GetMsg(window->UserPort);
522 XY ReplyMsg(message);
523 LG return(gadid);
524 Sr6 message = (struct IntuiMessage *)message->ExecMessag
e.mn_Node.ln_Succ;
525 Zq3 return (FALSE);
(C) 1988 M&T

```

Listing 5. (Schluß)

Programmname: sc4.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Bemerkung: siehe Listing 1

```

1 NmO /*****
*****
2 Sc6 Supercopy v1.0 (w) in 1988 by Eichenberger Software P
roduction
3 CpG loeschen und kopieren von Directories und ma
kedir
4 XZU TURBO-script creator
5 N70 Wed Oct 5 14:34:15 1988
6 Az *****/
7 rG /*=====
8 uV4 HEADERS
9 JB1 =====*/
10 Wz0 #include<functions.h>
11 d0 #include<vd0:sc_struct.h>
12 wL /*=====
13 Zw4 DEFINES
14 OG1 =====*/
15 Kx0 #define SOURCE OL
16 w6 #define DESTINATION 1L
17 lQ /*=====
18 YA4 GLOBALE VARIABLEN

```

```

19 TL1 =====*/
20 OUO extern struct Source src;
21 8o extern struct Destination dst;
22 F1 extern struct Volumes vlm;
23 EP extern struct GfxBase *GfxBase;
24 q1 extern struct RastPort *mainRP;
25 T2 extern struct Window *Main_Window;
26 T1 extern struct DosLibrary *DOSBase;
27 wV extern struct NewWindow rename_nw;
28 r4A struct Window *Rename_Window;
29 aBO extern struct NewWindow format_nw;
30 XzA struct Window *Format_Window;
31 IQO extern UBYTE MessageLine[80];
32 yc extern UBYTE r_m_buffer[80];
33 Uv extern UBYTE diskname_buffer[80];
34 XE extern struct Gadget MAIN_message;
35 mE extern struct Gadget rename_r_m;
36 ER extern struct IntuiText format_IText2;
37 qY static char *block_pfad[1];
38 yN /*****
*****
39 d5X Funktion fuer:
40 QqN Directory loeschen -> flag = 1
41 mD Directory kopieren -> flag = 0
42 4p0 Wed Oct 5 14:34:07 1988
43 8X *****/
44 BZ CopyDir(directory,flag,move)
45 EE2 char directory[];
46 NB3 ULONG flag;
47 kn BOOL move;
48 rW struct FileLock *dirlock,*oldlock;
49 U5 WORD success,check;
50 Ld register int i,t;
51 cC if((block_pfad[0][0] == NULL)){
52 L16 if(!(block_pfad[0]=AllocMem(300L,MEMF_CHIP) MEMF_CLEAR
)) }
53 rS9 if(!(block_pfad[1]=AllocMem(300L,MEMF_CHIP) MEMF_CLEAR
))){
54 6N WriteMessage("no memory free !");
55 fs3 namecat("",block_pfad[SOURCE],SOURCE);
56 en namecat(directory,block_pfad[SOURCE],3L);
57 9n namecat("",block_pfad[DESTINATION],DESTINATION);
58 l3 namecat(directory,block_pfad[DESTINATION],3L);
59 tH f(flag == 1){
60 4r6 if((dirlock=Lock(block_pfad[checkblocks()],ACCESS_REA
D))=0){
61 Jc9 WriteMessage("can't lock selected dir !");
62 TQ6 goto end;
63 eZ3 else{
64 996 if((dirlock=Lock(block_pfad[SOURCE],ACCESS_READ))=0)
{
65 ng9 WriteMessage("can't lock selected dir !");
66 XU6 goto end;
67 Y03 oldlock=CurrentDir(dirlock);
68 JN GetDir(dirlock,flag,move);
69 DM end;;
70 5e oldlock=CurrentDir(oldlock);
71 JT UnLock(dirlock);
72 qB if(flag == 1){
73 kQ6 if(DeleteFile(block_pfad[checkblocks()])=0){
74 D09 detect_errors();
75 Gr return(-1);
76 Xh3 block_pfad[0][0]=0;
77 lE FreeMem(block_pfad[0],300L);
78 pH FreeMem(block_pfad[1],300L);
79 zD0 GetDir(curr_lock,flag,move);
80 x52 struct FileLock *curr_lock;
81 wk3 ULONG flag;
82 JM BOOL move;
83 7T struct FileLock *newlock,*oldlock,*ignorelock,*dirlock;
84 Le struct FileInfoBlock *fib;
85 8R static BYTE loop;
86 5g WORD success,check;
87 wE register int i,t;
88 lL check=checkblocks();
89 8j if(flag=1) goto next;
90 Cb0 /*****
91 532 Directory im Destination-
92 4k3 Dir schon vorhanden ?
93 fX1 =====*/
94 3z3 if( ignorelock = Lock(block_pfad[DESTINATION],ACCESS_RE
AD)) == NULL){

```

Listing 6. »sc4.c« bitte mit dem Checksummer eingeben

```

95 616 WriteMessage("creating a new directory.");
96 Tu if(! (dirlock=CreateDir(block_pfad[DESTINATION]))){
97 aN9 detect_errors();
98 Au6 UnLock(dirlock);
99 m23 if(ignorelock != NULL) UnLock(ignorelock);
100 Ru0 next:
101 sM3 fib = (struct FileInfoBlock *)
102 lP9 AllocMem((long)sizeof(struct FileInfoBlock),MEMF_C
HIP|MEMF_CLEAR|MEMF_PUBLIC);
103 yW3 success=Examine(curr_lock,fib);
104 ez i=0;
105 zy if(flag==0) check=1;
106 ko2 while (success != 0)
107 x26 success = ExNext(curr_lock,fib); /* examine the n
ext entry */
108 5h if(success != NULL){
109 Jq9 if((fib->fib_EntryType > 0) && (success != NULL)
){
110 XaC namecat(&fib->fib_FileName[0],block_pfad[check
],3L);
111 Og newlock=Lock(&fib->fib_FileName[0]);/* neues D
irectory 'locken' */
112 Fj oldlock=CurrentDir(newlock); /* Director
y aktuell machen */
113 Un GetDir(newlock,flag,(BOOL)0);
114 Ek UnLock(newlock);
115 Xs if(flag == 1){
116 OQF if(DeleteFile(block_pfad[check])==0){
117 uhI detect_errors();
118 xY return(-1);
119 lAC oldlock=CurrentDir(curr_lock);
120 bL if(flag==1) Examine(curr_lock,fib); /* nur wenn
im Delete-Modi */
121 cw Parent_Dir(block_pfad[check],2L);
122 bW9 else{
123 EbC if(flag==0){
124 Z5F if(copy_file(block_pfad[check],&fib->fib_Fi
leName[0]) == -1){
125 URI goto end;
126 l3C if(flag == 1){
127 p9F if(fib->fib_EntryType <= 0){
128 mFI if(delete_files(block_pfad[check],&fib->
fib_FileName[0],fib,curr_lock)==-1){
129 YVC goto end;
130 Dm if(move == 1){
131 tDF if(fib->fib_EntryType <= 0){
132 yhI if(delete_files("",&fib->fib_FileName[0]
,fib,curr_lock)==-1){
133 cZ goto end;
134 wu9 i++;
135 fY0 end:
136 l23 FreeMem(fib,(long)sizeof(struct FileInfoBlock));
137 c80 copy_file(dst,source)
138 6n3 char dst[],source[];
139 D4 register struct FileHandle *fh1,*fh2;
140 Gs char filename[500],destination[500];
141 ig struct FileLock *oldlock;
142 2p char *file_buffer;
143 mn LONG error;
144 Qm LONG count,count1;
145 pv struct FileHandle *Open();
146 ZQ sprintf(destination,"%s/%s",dst,source);
147 ED WriteMessage(source);
148 d1 if((fh1 = Open(source,MODE_OLDFILE))==0){ /* source nam
e */
149 lu6 puts("can't open");
150 PP error = IoErr();
151 mM goto cleanup;
152 6b3 if((fh2 = Open(destination,MODE_NEWFILE))==0){ /* dst n
ame */
153 py6 puts("can't open");
154 TT error = IoErr();
155 qQ goto cleanup;
156 5i3 file_buffer = (char *)AllocMem(11264L,MEMF_CLEAR);
157 9g while(count=Read(fh1,file_buffer,5632L)){
158 Yv6 if(count == -1){
159 Jo9 WriteMessage("Read-Error - corrupt File !");
160 Re FreeMem(file_buffer,11264L);
161 Xm Close(fh2);Close(fh1);
162 FG return(-1);
163 EP6 if(Write(fh2,file_buffer,(LONG)count) != count){
164 jB9 WriteMessage("Write-Error");
165 WJ FreeMem(file_buffer,11264L);
166 cr Close(fh2);Close(fh1);
167 kL return(-1);

```

```

168 5b0 cleanup:
169 kt3 if(file_buffer) FreeMem(file_buffer,11264L);
170 Ca if(fh2) Close(fh2);
171 8U if(fh1) Close(fh1);
172 v00 } /* end of block */
173 l8 delete_files(dst,source,fib,lock)
174 gN3 char dst[],source[];
175 o7 struct FileInfoBlock *fib;
176 hK struct FileLock *lock;
177 rT char filename[500],destination[500];
178 GP register ULONG success;
179 l42 if(dst[0] != 0){
180 S16 switch(checkblocks()){
181 Mw9 case SOURCE:
182 whF if(source[0] != NULL)
183 68 if(DeleteFile(source)==0){
184 zmI detect_errors();
185 2d return(-1);
186 O3F success=Examine(lock,fib);
187 w5 break;
188 hN9 case DESTINATION:
189 3oF if(source[0] != NULL)
190 H8I sprintf(destination,"%s/%s",dst,source);
191 uhF else
192 QzI strcpy(destination,dst);
193 fqF if(DeleteFile(destination)==0){
194 9wI detect_errors();
195 Cn return(-1);
196 ADF success=Examine(lock,fib);
197 6F break;
198 fa6 return(0);
199 q13 else{
200 NP6 if(DeleteFile(source)==0){
201 G39 detect_errors();
202 Ju return(-1);
203 HK6 success=Examine(lock,fib);
204 lg return(0);
205 SX0 } /* end of block */
206 E6 write_turbo_script(){
207 Q13 FILE *fp;
208 Be char pfad[500];
209 Ls register int i;
210 sc fp=fopen("script.asc","a");
211 xG switch(checkblocks()){
212 rR6 case SOURCE:
213 V1 for(i=0;i<src.list.entrys;i++){
214 3zC if((src.list.select[i] == 1) && (src.list.dir[i
] != 1)){
215 2dF namecat(src.list.name[i],pfad,SOURCE);
216 Pd fprintf(fp,"%s\n",pfad);
217 Aq6 case DESTINATION:
218 FY for(i=0;i<dst.list.entrys;i++){
219 y8C if((dst.list.select[i] == 1) && (dst.list.dir[i
] != 1)){
220 mRF namecat(dst.list.name[i],pfad,DESTINATION);
221 U1 fprintf(fp,"%s\n",pfad);
222 dd3 }/*end of switch*/
223 jb fclose(fp);
(C) 1988 M&T

```

Listing 6. (Schluß)

Programmname: sc5.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Bemerkung: siehe Listing 1

```

1 Nm0 /*****
2 Sc6 Supercopy v1.0 (w) in 1988 by Eichenberger Software P
roduction
3 KdR Formatieren einer Diskette
4 En0 Sat May 14 17:15:41 1988
5 3w ****
*****SC3*/
6 qF /*****
7 tU4 HEADERS
8 IA1 ****
9 Vy0 #include <functions.h>
10 42 #include <devices/trackdisk.h>
11 d0 #include <vd0:sc_struct.h>
12 wL /*****
13 2w4 DEFINES
14 OG1 ****

```



```

15 Kx0 #define SOURCE OL
16 w6 #define DESTINATION 1L
17 lQ /*=====
18 YA4 GLOBALE VARIABLEN
19 TL1 =====*/
20 OU0 extern struct Source src;
21 8o extern struct Destination dst;
22 F1 extern struct Volumes vlm;
23 EP extern struct GfxBase *GfxBase;
24 q1 extern struct RastPort *mainRP;
25 Sk extern struct DosLibrary *DOSBase;
26 DL extern UBYTE MessageLine[80];
27 tX extern UBYTE r_m_buffer[80];
28 Pq extern UBYTE diskname_buffer[80];
29 S9 extern struct Gadget MAIN_message;
30 h9 extern struct Gadget rename_r_m;
31 c2 extern struct Image FRM_Image2;
32 JH extern struct Image FRM_Image9;
33 mt extern struct Image FRM_drive;
34 Qq extern struct Image FRM_format_rot;
35 Ys extern struct Image FRM_format_weiss;
36 Me extern struct Image verify_img;
37 Ce extern struct Border FRM_Border4;
38 cK extern struct NewWindow FRM_nw;
39 8h extern struct NewWindow rename_nw;
40 iH extern struct Window *Main_Window;
41 iAA struct Window *Format_Window;
42 5I struct Window *Rename_Window;
43 sr0 extern VOID drawbox();
44 So extern struct Gadget FRM_diskname;
45 wd #define FRM_ImageList1 FRM_Image2
46 BV #define FRM_BorderList1 FRM_Border4
47 WK #define NUMCYLS 80 /* normal # of cylind
ers */
48 1Y #define MAXCYLS (NUMCYLS+20) /* max # cyls to look
for during cal */
49 Nq #define NUMHEADS 2
50 4B #define NUMTRACKS (NUMCYLS*NUMHEADS)
51 JW #define BLOCKSIZE TD_SECTOR
52 hn #define CYLSIZE BLOCKSIZE*NUMHEADS*NUMSECS
53 py #define MOTOR_ON 1
54 cE #define MOTOR_OFF 0
55 EI struct MsgPort *diskport;
56 lk struct IOStdReq *diskreq;
57 n2 ULONG openererror = -999;
58 QL ULONG Block;
59 hF char mask[]={1,4,9};/**/
60 uR unsigned char *disk_memory,*abfall_memory;
61 ae ULONG *buf;
62 M1 /*=====
63 9NH Format Requester init. und auf Aktionen wa
rten
64 Ts0 /*=====
65 nV InitFormatReq(){
66 KW3 register LONG success;
67 o82 struct IntuiMessage *Message;
68 U9 register WORD mouse_x,mouse_y;
69 Y23 register struct RastPort *rp,*Open_FormatWindow();
70 3I struct IntuiMessage *Wait_for_Message();
71 zB BOOL verify_flag=1;
72 aT2 LONG drive=0,test_drive,Check_connected_Drives();
73 9V3 LONG gadid;
74 JL ULONG count;
75 JD rp=Open_FormatWindow();
76 k02 test_drive=Check_connected_Drives();
77 Fi3 ActivateGadget(&FRM_diskname,Format_Window,NULL);
78 gq FOREVER{
79 lr6 Message=Wait_for_Message(Format_Window,1L);
80 DY5 gadid = (USHORT)((struct Gadget *) Message->IAddress)
->GadgetID;
81 6j6 if(gadid == CANCEL){
82 Nf CloseWindow(Format_Window);
83 029 return(FALSE);
84 ct6 if(Message->Class == RAWKEY){
85 H09 switch(Message->Code){
86 hwC case 0x34: /* fuer 'V' */
87 cdF if(verify_flag == 1){
88 FeI verify_flag=0;
89 Xb DrawImage(rp,&verify_img,0L,0L);
90 50E else{
91 MmI verify_flag=1;
92 Gs RectFill(rp,232L,52L,(long)232+30,(long)5
2+14);

```

```

93 QZF break;
94 A16 if((gadid == OK) || (gadid == DISKNAME)){
95 s59 FormatingDisk(drive,diskname_buffer,verify_flag,rp
);
96 bt6 CloseWindow(Format_Window);
97 KQ9 /*FormatingDisk(drive,diskname_buffer,verify_flag,
rp);*/
98 Jp goto quit;
99 J86 if(Message->Class == MOUSEBUTTONS){
100 159 mouse_x=Message->MouseX;
101 qF mouse_y=Message->MouseY;
102 9Y if((mouse_x >= 75) && (mouse_x <= 158) &&
103 EFC (mouse_y >= 36) && (mouse_y <= 45)){
104 YUA count = (mouse_x-75)/28; /* 0, 1, 2 */
105 DIC if((test_drive&mask[count])==0) goto weiter; /*
*/
106 daA SetAPen(rp,0L);
107 TMC RectFill(rp,80L,25L,170L,35L); /* aktuelles dri
ve loeschen*/
108 ry DrawImage(rp,&FRM_drive,(long)(count*30),0L);
109 wQ drive=count;
110 zK6 weiter:
111 W30 quit:
112 AZ /*=====
113 6PR Formatieren einer Diskette
114 Hg0 /*=====
115 3J FormatingDisk(Drive,diskname,verify_flag,rp)
116 Od3 ULONG Drive;
117 pe UBYTE diskname[];
118 Np2 BOOL verify_flag;
119 DT3 struct RastPort *rp;
120 qW register ULONG i,n;
121 vh2 char text[3];
122 ht3 LONG Wait_on_Message();
123 1I Open_All(Drive);
124 Rp SwitchMotor(MOTOR_ON);
125 gn for(i=0;i<NUMCYLS;i++){
126 Oy5 SetAPen(rp,1L);
127 Qf6 SetDrMd(rp,JAM2);
128 Ew sprintf(text,"%ld",i);
129 An Move(rp,200L,63L);
130 aT Text(rp,text,(long)strlen(text));
131 KL if(verify_flag == 1){
132 kG9 DrawImage(rp,&FRM_format_rot,0L,0L);
133 G0 FormatCylinder(i,disk_memory);
134 7y if(Wait_on_Message(Format_Window) == CANCEL){
135 ebC goto end;
136 mk9 DrawImage(rp,&FRM_format_weiss,0L,0L);
137 Lz DrawImage(rp,&FRM_format_rot,0L,8L);
138 GS if(Read_DiskBlock(i,abfall_memory,(BOOL)1)==-1){
139 aGC puts("Verify_Error !!!");
140 c49 DrawImage(rp,&FRM_format_weiss,0L,8L);
141 up6 else{
142 uQ9 DrawImage(rp,&FRM_format_rot,0L,0L);
143 QA FormatCylinder(i,disk_memory);
144 H8 if(Wait_on_Message(Format_Window) == CANCEL){
145 olC goto end;
146 wu9 DrawImage(rp,&FRM_format_weiss,0L,0L);
147 Io3 Write_DiskInformation(diskname);
148 sl0 end:
149 s03 SwitchMotor(MOTOR_OFF);
150 mK cleanup(NULL);
151 Ba0 /*=====
152 2Y4 alles freigeben
153 dV1 /*=====
154 hc0 cleanup(mes)
155 lU char *mes;
156 BL3 if (mes) puts(mes);
157 wV /* drive df0: wieder freigeben */
158 Cx if(!openererror) CloseDevice(diskreq);
159 cr if(diskreq) DeleteStdIO(diskreq);
160 F6 if(diskport) DeletePort(diskport);
161 2D if(disk_memory) FreeMem(disk_memory, (ULONG)CYLSIZE);
162 7o if(abfall_memory) FreeMem(abfall_memory, (ULONG)CYLSIZE);
;
163 Nm0 /*=====
164 So4 Drive ein/aus
165 ph1 /*=====
166 z10 SwitchMotor(onoffswitch)
167 LU ULONG onoffswitch;
168 ti3 diskreq->io_Length = onoffswitch;

```

Listing 7. »sc5.c« bitte mit dem Checksummer eingeben

```

169 xo      diskreq->io_Command = TD_MOTOR;
170 uC      DoIO(diskreq);
171 w70 /* lesen eines Blocks von Diskette */
172 Eh Read_DiskBlock(block, buf, flag)
173 pG      ULONG block;
174 pQ      unsigned char *buf;
175 Fp      BOOL flag;
176 1k3      diskreq->io_Length = BLOCKSIZE;
177 VV      if(flag == 1) diskreq->io_Length = CYLSIZE;
178 1e      diskreq->io_Data = (APTR) buf;
179 Pe      diskreq->io_Command = CMD_READ;
180 nC      diskreq->io_Offset = block*BLOCKSIZE;
181 ct      if(flag == 1) diskreq->io_Offset = block*CYLSIZE;
182 60      DoIO(diskreq);
183 jG      if(diskreq->io_Error != 0){
184 nt6      puts(" *** Read-Error ***");
185 z1      printf("-- Block nr. %d --- Error Number is: %d\n",
          block, diskreq->io_Error);
          return(-1);
186 3e      return(0);
187 UP3
188 mB0 /*=====
189 Q44 schreiben eines Blocks
190 E61 =====*/
191 L20 Write_DiskBlock(block, buf)
192 Is2      int block;
193 AS3      UBYTE *buf;
194 J2      diskreq->io_Length = BLOCKSIZE;
195 Iv      diskreq->io_Data = (APTR) buf;
196 ao      diskreq->io_Command = CMD_WRITE;
197 4T      diskreq->io_Offset = block*BLOCKSIZE;
198 Me      DoIO(diskreq);
199 zw      if(diskreq->io_Error != 0){
200 Ws6      puts(" *** Write-Error ***");
201 F1      printf("-- Block nr. %d --- Error Number is: %d\n",
          block, diskreq->io_Error);
202 OP0 /*=====
203 Q14 Updaten eines Blocks
204 SK1 =====*/
205 mW0 Update()
206 z83      diskreq->io_Command = CMD_UPDATE;
207 Vn      DoIO(diskreq);
208 85      if(diskreq->io_Error != 0){
209 AC6      puts(" *** Update-Error ***");
210 hZ      printf("Error Number is: %ld\n", diskreq->io_Error);
211 9Y0 /*=====
212 XC1 Track formatieren -> ohne Verify
213 bT      =====*/
214 690 FormatCylinder(cyl, buf)
215 pq3      LONG cyl;
216 Xp      UBYTE *buf;
217 wH      diskreq->io_Length = CYLSIZE;
218 fI      diskreq->io_Data = (APTR) buf;
219 iK      diskreq->io_Command = TD_FORMAT;
220 fd      diskreq->io_Offset = cyl*CYLSIZE;
221 j1      DoIO(diskreq);
222 MJ      if(diskreq->io_Error != 0){
223 G06      puts(" *** FORMAT-Error ***");
224 vn      printf("Error Number is: %ld\n", diskreq->io_Error);
225 Nm0 /*=====
226 8Z Track formatieren -> mit Verify
227 ph1 =====*/
228 zh0 Verify_Format(cyl, buf)
229 343      LONG cyl;
230 l3      UBYTE *buf;
231 AV      diskreq->io_Length = CYLSIZE;
232 tW      diskreq->io_Data = (APTR) buf;
233 j9      diskreq->io_Command = ETD_FORMAT;
234 tr      diskreq->io_Offset = cyl*CYLSIZE;
235 xF      DoIO(diskreq);
236 aX      if(diskreq->io_Error != 0){
237 UE6      puts(" *** FORMAT-Error ***");
238 91      printf("Error Number is: %ld\n", diskreq->io_Error);
239 b00 /*=====
240 aD2 Pruefsumme neu berechnen
241 3v1 =====*/
242 4v0 CorrectBlockChecksum(buf)
243 yG      UBYTE *buf;
244 uS3      ULONG checksum, *long_ptr;
245 Qs      LONG i;
246 jH      long_ptr = (ULONG *)buf;
247 pC      checksum = 0;
248 8n      for(i=0; i< BLOCKSIZE/4; i++)
249 7x6      checksum += long_ptr[i];

```

```

250 jM3      long_ptr[5] -= checksum;
251 nC0 /*=====
252 Qx4      Block besetzen
253 F71      =====*/
254 190 Alloc_DiskBlock(block, buf)
255 Jt      int block;
256 BT      UBYTE *buf;
257 s23      ULONG *long_ptr = (ULONG *) buf;
258 OQ      LONG longword, bit, i;
259 DD      ULONG checksum;
260 T0      longword = (block-2)/32;
261 x5      bit = block-2-longword*32;
262 SU      long_ptr[longword+1] &= **B(1L<<bit);
263 b9      /* BAM checksumme berechnen */
264 pH      for(i=1, checksum = 0; i<BLOCKSIZE/4; i++)
265 ND6      checksum += long_ptr[i];
266 Sm3      long_ptr[0] = -checksum;
267 3S0 /*=====
268 Jo4      coeffne Device, Port usw.
269 VN1      =====*/
270 c70 Open_All(drive)
271 XL3      LONG drive;
272 xS      /* CHIP Memory fuer buf holen */
273 zk5      if(! (disk_memory = AllocMem((LONG)CYLSIZE, MEMF_CHIP |
          MF_CLEAR)))
274 NuD      cleanup("No memory !");
275 2U5      if(! (abfall_memory = AllocMem( (LONG)CYLSIZE, MEMF_CHIP
          | MEMF_CLEAR ) ) )
276 PwD      cleanup("No memory !");
277 IZ3 /* Port zur Kommunikation mit trackdisk.device aufbauen
278 Gt5      if(! (diskport = CreatePort("mydisk.port", 0L)))
279 xK8      cleanup("CreatePort failed!");
280 cH3 /* StdIO-Block zur Kommunikation ueber 'diskport' aufbau
          en */
          if(! (diskreq = CreateStdIO(diskport)))
281 KyA      cleanup("CreateStdIO failed!");
282 C38
283 vv3 /* reservieren der trackdisk.device df0: */
284 TR5      openererror = OpenDevice(TD_NAME, (LONG)drive, diskreq, dri
          ve);
          if(openererror) cleanup("OpenDevice failed!");
285 H98
286 M10 /*=====
287 k24      mache Disk Doswuerdig!!
288 og1      =====*/
289 HMO Write_DiskInformation(diskname)
290 cs3      char diskname[];
291 bH      register ULONG i, n;
292 nQ      ULONG *l_buf;
293 Fo      ULONG namelength;
294 m0      for(i=0; i<4; i++)
295 SG6      disk_memory[i] = "DOS\0[i];
296 m53      Write_DiskBlock(0, disk_memory);
297 7b      l_buf = (ULONG *)disk_memory;
298 ER      l_buf[0] = 2;
299 Er      l_buf[3] = 72;
300 Wh      l_buf[78] = 0xffffffff;
301 jB      l_buf[79] = 881;
302 5L      DateStamp(&l_buf[121]);
303 KI      DateStamp(&l_buf[105]);
304 Ev      disk_memory[432] = strlen(diskname);
305 vN      for(i=0; i<strlen(diskname); i++)
306 B06      disk_memory[433+i] = diskname[i];
307 NU3      l_buf[127] = 1;
308 fA      CorrectBlockChecksum(disk_memory);
309 eF      Write_DiskBlock(880, disk_memory);
310 P1      for(i=1; i<56; i++)
311 br6      l_buf[i] = 0xffffffff;
312 IX3      Alloc_DiskBlock(880, disk_memory);
313 Oe      Alloc_DiskBlock(881, disk_memory);
314 oQ      Write_DiskBlock(881, disk_memory);
315 Ju      Update();
316 Sr0 /*=====
          =====
317 u7N      Format Window oeffnen und zeichnen
318 Ao0 Sun May 15 10:24:39 1988
319 az      =====
          =====
320 WK      struct RastPort *Open_FormatWindow(){
321 EU2      register struct RastPort *rp;
322 X33      register WORD i;
323 6q      static LONG y1[] = { 8, 10, 12, 14, 16 };
324 dF      static LONG y2[] = { 78, 80, 82, 84, 86 };
325 Ou2      Format_Window=OpenWindow((struct NewWindow *) &FRM_nw);

```



```

326 vu3 rp = Format_Window->RPort;
327 FD SetAPen(rp,1L);
328 1T for(i=0;i<5;i++){
329 FK6 Move(rp,9L,(long)y1[i]);
330 tH Draw(rp,355L,(long)y1[i]);
331 rH Move(rp,63L,(long)y2[i]);
332 N3 Draw(rp,302L,(long)y2[i]);
333 AJ3 Move(rp,1L,22L);
334 p1 Draw(rp,365L,22L);
335 cI Move(rp,1L,47L);
336 VP Draw(rp,365L,47L);
337 VV SetAPen(rp,3L);
338 ta for(i=0;i<4;i++){
339 jn6 drawbox(Format_Window->RPort,(long)187+i,(long)51+i,
(long)187+38-1,(long)51+18-1);
340 cQ3 DrawBorder(rp,&FRM_BorderList1,0L,0L);
341 hI4 DrawImage(rp,&FRM_ImageList1,0L,0L);
342 Ah DrawImage(rp,&FRM_Image9,0L,0L);
343 x43 DrawImage(rp,&FRM_drive,0L,0L); /* */
344 8r return(rp);
345 YJO /* last Update:
346 kT Wed Oct 5 14:34:41 1988
347 e3 */
348 x9 LONG Check_connected_Drives(){
349 r13 register LONG i,n=0,count=0;
350 C7 static char *drives[] = { "DF0", "DF1", "DF2" };
351 Gn for(i=0;i<=29;i++){
352 GE6 for(n=0;n<=2;n++){
353 ym9 if(strcmp(vlm.n[i],drives[n]) == 0){
354 snC count+=mask[n];/**/
355 5P3 return(count);
(C) 1988 M&T

```

Listing 7. (Schluß)

Programmname: sc6.c

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: C

Bemerkung: siehe Listing 1

Programmname: sc6.c

```

1 Nm0 /*****
*****
2 Sc3 Supercopy v1.0 (w) in 1988 by Eichenberger Software Prod
uction
3 KPC 1. Zeigt die freie Speicherkapazitaet eines Lau
ferwerks an
4 dT 2. Type und Typenh
5 7WI 3. Command
6 Bt0 Wed Oct 5 14:35:12 1988
7 Yx *****/
*****
8 sH /*=====
9 vW4 HEADERS
10 KC1 =====*/
11 X00 #include<functions.h>
12 e1 #include<vd0:sc_struct.h>
13 xM /*=====
14 3x4 DEFINES
15 PH1 =====*/
16 nU0 #define SPACE 0x20
17 qW #define BACKSPACE 0x08
18 iY #define LINEFEED 0x0a
19 nX #define RETURN 0x0d
20 We #define CSI 0x9b
21 a1 #define DEVICE_STATUS_REPORT 0x36, 0x6e
22 OS #define WINDOW_STATUS_REQUEST 0x30, 0x20, 0x71
23 oX #define DELETE_LINE 0x4d
24 Sg #define CURSOR_UP 0x41
25 Dn #define CURSOR_DOWN 0x42
26 4F #define CURSOR_Shift_UP 0x54
27 On #define CURSOR_Shift_DOWN 0x53

```

```

28 FP #define ARG1 *(long_memory+counter+1),*(long_memory+counte
r+2),*(long_memory+counter+3),*(long_memory+counter+4)
29 YB #define SOURCE 0L
30 AK #define DESTINATION 1L
31 Lv UBYTE del_line[] = { CSI,DELETE_LINE,CSI,0x32,CURSOR_UP };
32 y1 UBYTE change_color[] = { CSI,'2',';', '3','3',';', '4','0','m'
,
33 CNO CSI,'0',';', '3','1',';', '4','0','m'
};
34 Ih0 /*=====
35 pR4 GLOBALE VARIABLEN
36 kc1 =====*/
37 H10 extern struct Source src;
38 P5 extern struct Destination dst;
39 WI extern struct Volumes vlm;
40 Vg extern struct GfxBase *GfxBase;
41 7z extern struct RastPort *mainRP;
42 kJ extern struct Window *Main_Window;
43 K2 extern struct DosLibrary *DOSBase;
44 Dm extern struct NewWindow rename_nw;
45 8LA struct Window *Rename_Window;
46 rS0 extern struct NewWindow format_nw;
47 oGA struct Window *Format_Window;
48 pB7 struct Window *Type_Window;
49 a10 extern UBYTE MessageLine[80];
50 Gu extern UBYTE r_m_buffer[80];
51 mD extern UBYTE diskname_buffer[80];
52 pW extern struct Gadget MAIN_message;
53 4W extern struct Gadget rename_r_m;
54 WJ extern struct IntuiText format_iText2;
55 P4 struct IntuiMessage *message;
56 e3 /*=====
57 6C4 Globale Variablen fuer Type-Window
58 6yl =====*/
59 iV0 struct MsgPort *Con_w_Port;
60 ZH struct MsgPort *Con_r_Port;
61 IL struct IOStdReq *Con_w_Msg;
62 42 struct IOStdReq *Con_r_Msg;
63 wn ULONG conErr;
64 L2 struct NewWindow Type_nw = { 0,0,640,256,0,1,GADGETUP+MOUSEB
UTTONS+CLOSEWINDOW+RAWKEY,
65 ox3 WINDOWCLOSE+WINDOWSIZING+WINDOWDEPTH+ACTIVATE,NULL,NULL,
NULL,NULL,NULL,640,20,640,257,
66 1K WBENCHSCREEN
67 OK0 };
68 xv struct FileHandle *filehandle=0;
69 Ts /*****
*****
70 lv6 schreibt die aktuelle Diskettenkapazitaet in den 'fre
e-space' block
71 uEM wenn call = 0 => die zahl addieren
72 uE wenn call = 1 => die zahl subtrahie
ren
73 eE wenn call = 2 => aktualisieren
74 d20 *****/
*****
75 ol Write_Info(flag,call,zahl)
76 rf3 ULONG flag;
77 91 BOOL call;
78 Cw ULONG zahl;
79 p1 static ULONG actual_diskspace[2];
80 rL struct InfoData *infodata;
81 An4 struct FileLock *lock;
82 ed2 char bytes[12];
83 uk3 LONG count;
84 k12 infodata = (struct InfoData *) AllocMem((long)sizeof(stru
ct InfoData),MEMF_CLEAR);
85 v03 SetAPen(mainRP,1L);
86 pJ2 SetDrMd(mainRP,JAM2);
87 Vf3 switch(flag){
88 rR6 case SOURCE:
89 in9 lock =Lock(src.spath,ACCESS_READ);
90 JR Info(lock,infodata);
91 Y7 if(strcmp(src.spath,"RAM:",4) != NULL){
92 G7 count=(infodata->id_NumBlocks-infodata->id_NumBl
ocksUsed)
93 MMG * (infodata->id_BytesPerBlock);
94 L89 else
95 IWC count = AvailMem(MEMF_CHIP)+AvailMem(MEMF_FAST)
;
96 n66 switch(call){

```

Listing 8. »sc6.c« bitte mit dem Checksummer eingeben

```

97 ZpC         case 0:
98 1NF             count+=zahl;break;
99 duC         case 1:
100 7VF             count-=zahl;break;
101 or9             if(count>99999999) count==0; /**/
102 xL             sprintf(bytes,"%8ld",count);
103 YH             Move(mainRP,182L,191L);
104 ye             Text(mainRP,bytes,(long)strlen(bytes));
105 278             Unlock(lock);
106 dm6             break;
107 04             case DESTINATION:
108 F89                 lock =Lock(dst.dpath,ACCESS_READ);
109 ck                 Info(lock,infodata);
110 5S                 if(strncmp(dst.dpath,"RAM:",4) != NULL){
111 ZQ                     count=(infodata->id_NumBlocks-infodata->id_NumBl
ocksUsed)
                        * (infodata->id_BytesPerBlock);
                        else
112 ffG                 count = AvailMem(MEMF_CHIP)+AvailMem(MEMF_FAST)
113 eR9
114 bpC
115 6P6             switch(call){
116 s8C                 case 0:
117 KgF                     count+=zahl;break;
118 wDC                 case 1:
119 QoF                     count-=zahl;break;
120 LR9                     if(count < 0) count = 0;
121 Ge                     sprintf(bytes,"%8ld",count);
122 7H                     Move(mainRP,182+364L,191L);
123 Hx                     Text(mainRP,bytes,(long)strlen(bytes));
124 LQ                     Unlock(lock);
125 w56                 break;
126 Av3                 FreeMem(infodata,(long)sizeof(struct InfoData));
127 Po0             /*****
*****
128 aic                                     TYPE
129 po0 Tue May 17 19:00:55 1988
130 Xw *****
*****
131 I4             Type(){
132 012             struct IntuiMessage *Message,*Wait_for_Message();
133 Y83             register LONG gadid;
134 04             ULONG i=0,n=0,q;
135 dH             UBYTE *mybuffer=0,*Open_type_things();
136 XI             char Read_Buffer[80];
137 oM2             char filename[180];
138 V53             UBYTE zeilen_speicher[180];
139 Qu             LONG laenge,z=0,FileSize();
140 o1             Wait_for_Click(Read_Buffer,filename,&laenge);
141 6Z0 next:
142 263             mybuffer=Open_type_things(filename,&laenge);
143 u5             while((i < laenge)){
144 RR6                 while((*mybuffer+i) != '\n') && (n<180)){
145 LP9                     zeilen_speicher[n++]=(*mybuffer+i++);
146 EC6                     if(n>136){
147 es9                         WriteMessage("incorrect. Use HTYPE !");
148 CO                         Close_type_things(&laenge,mybuffer);
149 S3                         return(-1);
150 816                     zeilen_speicher[n++]='\n';
151 Qd                     zeilen_speicher[n++]=NULL;
152 a0                     n=0;
153 FD                     i++;
154 sn                     ConsoleStringWrite(Con_w_Msg,zeilen_speicher);
155 p4                     z++;
156 He                     q=(Type_Window->Height-27)/8;
157 d2                     if(z==q){
158 JX9                         if(Wait_for_Space((UWORD)0) == 0) goto end2;
159 5h                         z=0;
160 fd4                     }/*if*/
161 hq6                     end;
162 y30 end2:
163 wc3             Wait_for_Space((UWORD)1);
164 mG             WriteMessage("");
165 Tf             Close_type_things(&laenge,mybuffer);
166 2R0             /*****
*****
167 dQc                                     HTYPE
168 V60 Sun May 22 13:53:33 1988
169 AZ *****
*****
170 zU             HType(){
171 Uo2             struct IntuiMessage *Message;
172 B13             register LONG gadid;
173 ZH             ULONG i=0,n=0,q,buf=0,counter=0;
174 EQ2             ULONG *long_memory;
175 FD3             UBYTE *mybuffer=0;

```

```

176 Bw             .char Read_Buffer[80];
177 S02             char filename[180];
178 9J3             UBYTE zeilen_speicher[180];
179 4Y             LONG laenge,z=0,FileSize();
180 SP             Wait_for_Click(Read_Buffer,filename,&laenge);
181 kD0 next:
182 gk3             mybuffer=Open_type_things(filename,&laenge);
183 Os0             /*****/
184 I7             /*****
185 dN             n = Zaehler fuer 'zeilen_speicher' (UBYTE)
186 PA             buf = Zaehler fuer 'mybuffer' (UBYTE)
187 B31             =====
188 3D3             long_memory = (ULONG *)mybuffer;
189 ep             while((i < laenge)){
190 n16                 sprintf(&zeilen_speicher[n],"%04ld: %08lx %08lx %08lx
%08lx %08lx ]",i*4,*(long_memory+counter),ARG1);
191 Jp                 n+=52;
192 di                 counter += 5;
193 lg                 for(q=0;q<20;q++){
194 xr9                     if((*mybuffer+buf) <= 31) ] (*mybuffer+buf) >
= 128){
195 GxC                         sprintf(&zeilen_speicher[n],".");
196 kD                         buf++;
197 7A                         n++;
198 pk9                     else{
199 l3C                         sprintf(&zeilen_speicher[n],"%c",*(mybuffer+buf
));
200 oH                         buf++;
201 BE                         n++;
202 w16                         sprintf(&zeilen_speicher[n],"]");n++;
203 9t                         sprintf(&zeilen_speicher[n],"\n");n++;
204 Mm                         i+=5;
205 IV                         zeilen_speicher[n++]=NULL;
206 Ss                         n=0;
207 75                         i++;
208 kf                         ConsoleStringWrite(Con_w_Msg,zeilen_speicher);
209 hw                         z++;
210 9W                         q=(Type_Window->Height-27)/8;
211 Vu                         if(z==q){
212 kD9                             if(Wait_for_Space((UWORD)0) == 0) goto weiter;
213 x2                             z=0;
214 XV4                             }/*if*/
215 g10 weiter:
216 nT3             Wait_for_Space((UWORD)1);
217 d7             WriteMessage("");
218 KW             Close_type_things(&laenge,mybuffer);
219 Hg0             /*****
220 Oc2             eroeffnet alles wichtige
221 jb1             =====
222 xq0 Open_all_Con_things(){
223 iA3             if(!(Con_w_Port = CreatePort("ESP-con-dev_write",0L)))
224 yM4                 puts("I can't open the console-writeport !!");
225 Bs6                 kill_all_type();
226 Wz3             if(!(Con_w_Msg = CreateStdIO(Con_w_Port))){
227 7K6                 puts("I can't open the console-write-messageport !!");
228 Ev
229 mU3                 kill_all_type();
230 r44             if(!(Con_r_Port = CreatePort("ESP-con-dev_read",0L))){
231 Hy6                 puts("I can't open the console-readport !!");
232 el3                 kill_all_type();
233 QD6             if(!(Con_r_Msg = CreateStdIO(Con_r_Port))){
234 K1                 puts("I can't open the console-read-messageport !!");
235 qy3                 kill_all_type();
236 p86             if(!(Type_Window = OpenWindow(&Type_nw))){
237 N4                 puts("I can't open my high quality spezial Window!!!!
!!!");
238 S23                 kill_all_type();
239 Zp                 Con_w_Msg->io_Data = (APTR)Type_Window;
240 8a2                 Con_w_Msg->io_Length = sizeof(*Type_Window);
if((conErr = OpenDevice("console.device",0L,Con_w_Msg,0L)
)!=NULL){
241 WS6                     puts("I can't open the console.device");
242 S9                     kill_all_type();
243 lL3                 Con_r_Msg->io_Device = Con_w_Msg->io_Device;
244 4H                 Con_r_Msg->io_Unit = Con_w_Msg->io_Unit;
245 h60             /*****
246 8a4             alles zurueckgeben
247 911             =====
248 cLO kill_all_type(){
249 c53             if(!conErr) CloseDevice(Con_w_Msg);
250 Ow             if(Con_w_Port) DeletePort(Con_w_Port);
251 vJ             if(Con_r_Port) DeletePort(Con_r_Port);
252 HB             if(Con_w_Msg) DeleteStdIO(Con_w_Msg);
253 JT             if(Con_r_Msg) DeleteStdIO(Con_r_Msg);
254 8r             if(Type_Window) CloseWindow(Type_Window);

```



```

255 rG0 /*=====
256 lU4 auf Con schreiben
257 JB1 =====*/
258 9U0 ConsoleWrite(StdReq,string,length)
259 k33 struct IOStdReq *StdReq;
260 P5 UBYTE string[];
261 nJ ULONG length;
262 Nn StdReq->io_Command = CMD_WRITE;
263 9j StdReq->io_Data = (APTR)string;
264 mc StdReq->io_Length = length;
265 4L DoIO(StdReq);
266 2R0 /*=====
267 4G4 auf Con String schreiben
268 UM1 =====*/
269 e40 ConsoleStringWrite(StdReq,string)
270 vE3 struct IOStdReq *StdReq;
271 aG UBYTE string[];
272 Xx StdReq->io_Command = CMD_WRITE;
273 Jt StdReq->io_Data = (APTR)string;
274 bv StdReq->io_Length = -1L;
275 EV DoIO(StdReq);
276 Cb0 /*=====
277 c84 Zeichen lesen
278 eW1 =====*/
279 470 QueueRead(StdReq, buffer, length)
280 50 struct IOStdReq *StdReq;
281 Q7 char *buffer;
282 63 ULONG length;
283 HE3 StdReq->io_Command = CMD_READ;
284 Mx StdReq->io_Data = (APTR)buffer;
285 7x StdReq->io_Length = length;
286 96 SendIO(StdReq);
287 z00 /*=====
*****
288 YBT Warte auf Space-Taste
289 2z0 Sun May 22 10:06:35 1988
290 7W *****
*****
291 hX Wait_for_Space(flag)
292 u93 UWORD flag;
293 Fk ULONG MessageClass;
294 cp USHORT code;
295 2s2 struct IntuiMessage *message;
296 613 ConsoleWrite(Con_w_Msg,change_color,9L);
297 gX if(flag ==0)
298 626 ConsoleStringWrite(Con_w_Msg, "\n*** Please press SPAC
E to continue.***");
299 eR2 else
300 t16 ConsoleStringWrite(Con_w_Msg, "\n*** END OF FILE.***")
;
301 dM3 ConsoleWrite(Con_w_Msg,&change_color[9],9L);
302 IS2 FOREVER{
303 xe4 if (message = (struct IntuiMessage *)
304 Hr8 GetMsg(Type_Window->UserPort)){
305 599 MessageClass = message->Class;
306 0d code = message->Code;
307 45 ReplyMsg(message);
308 g3 switch (MessageClass){
309 p18 case CLOSEWINDOW :
310 TOC return(0);
311 w5 break;
312 AC case RAWKEY:
313 QMI switch(code){
314 eKL case 64:
315 TOI ConsoleWrite(Con_w_Msg,&del_line[0],2L);
316 hH ConsoleWrite(Con_w_Msg,&del_line[2],3L);
317 6BR ConsoleStringWrite(Con_w_Msg, "\n
");
318 bY goto end;
319 4D break;
320 fuE case 0x45:
321 eZR return(0);
322 7GI break;
323 8H break;
324 JH4 }/*if*/
325 zy3 }/*forever*/
326 MV end;
327 ok return(1);
328 e30 /*=====
*****
329 kKt Filegroesse bestimmen
330 jU0 Sun May 22 10:23:25 1988
331 mB *****
*****
332 KJ LONG FileSize (name)

```

```

333 Ot3 char name[];
334 ST BPTR lock;
335 O3 struct FileInfoBlock *fileinfoblock;
336 hI fileinfoblock = AllocMem ((LONG)sizeof (struct FileInfoB
lock),MEMF_CLEAR);
337 r4 lock = Lock (name, ACCESS_READ);
338 wR if (lock == NULL) return (-1);
339 mB if (Examine (lock,fileinfoblock) == FALSE) return (-1);
340 v2 UnLock (lock);
341 fP return (fileinfoblock->fib_Size);
342 sH0 /*=====
*****
343 pEP Warte bis Name angeklickt wird
344 TNO Sun May 22 11:27:29 1988
345 OP *****
*****
346 TL Wait_for_Click(Read_Buffer,filename,laenge)
347 aM3 char *Read_Buffer;
348 6s char *filename;
349 sM ULONG *laenge;
350 WH2 struct IntuiMessage *Message,*Wait_for_Message();
351 G83 BOOL Block_Control1();
352 Ib register LONG gadid,FileSize(),err;
353 M1 WriteMessage("Please select a name to typing.");
354 8I FOREVER{
355 i16 Message=Wait_for_Message(Main_Window,0);
356 ax if(Message == -1) return(-1);
357 Ir5 gadid = ((struct Gadget *) Message->IAddress)->Gadge
tID;
358 rQ6 if(Block_Control1(gadid) == 1) continue;
359 Le switch(checkblocks()){
360 Fp case SOURCE:
361 SUC if( (gadid >=3) && (gadid <=15) ){
362 nvF strcpy(Read_Buffer,src.list.name[gadid-3+src
.scroll_count]);
363 g8 goto next;
364 hUC else
365 EnF err=1;
366 pyC break;
367 aG9 case DESTINATION:
368 DZC if( (gadid >= 48) && (gadid <=61) ){
369 BzF strcpy(Read_Buffer,dst.list.name[gadid-48+ds
t.scroll_count]);
370 nF goto next;
371 obC else
372 LuF err=1;
373 w5C break;
374 o39 default: err=1;break;
375 W66 if(err==1) continue;
376 tM0 next:
377 TH3 namecat(Read_Buffer,filename,(long)checkblocks());
378 l1 *laenge = FileSize(filename);
379 Ts0 /*=====
*****
380 KyB Oeffne alle console.device Sachen. Lese File in
den Buffer
381 ue0 Sun May 22 11:33:09 1988
382 b0 *****
*****
383 vr UBYTE *Open_type_things(filename,laenge)
384 GL3 char filename[];
385 Sw ULONG *laenge;
386 Ww static UBYTE *buffer;
387 WN Open_all_Con_things();
388 oX buffer = (UBYTE *)AllocMem(*laenge,MEMF_CLEAR|MEMF_CHIP)
;
389 Jo filehandle = Open(filename,MODE_OLDFILE);
390 4c Read(filehandle,buffer,*laenge);
391 Wf return(buffer);
392 g50 /*=====
*****
393 PDD Schliesse alle console.device Sachen. Schliess
e File.
394 6r0 Sun May 22 11:35:08 1988
395 oD *****
*****
396 eJ Close_type_things(laenge,mybuffer)
397 tw3 LONG *laenge;
398 GV UBYTE *mybuffer;
399 YP if(filehandle != NULL) Close(filehandle);
400 Oh kill_all_type();
401 m6 if(mybuffer != NULL) FreeMem(mybuffer,*laenge);
(C) 1988 M&T

```

Listing 8. (Schluß)

Auch in diesem Sonderheft gibt's wieder eine geballte Ladung Tips und Tricks für Sie. Wir haben wieder einmal kräftig in unserer Trickkiste gewählt. Aber was heißt hier in unserer Trickkiste — natürlich in der unserer Leser. Denn wir bekommen laufend die heißesten Kniffe zugesandt und wollen diese natürlich für alle Leser zugänglich machen.

Sicher haben auch Sie schon mit dem einen oder anderen Kunstgriff dem Amiga auf die Sprünge geholfen. Behalten Sie aber Ihre Kniffe nicht für sich, sondern machen Sie diese auch für den Rest der Leserschaft zugänglich, indem Sie sie an uns schicken. Veröffentlichte Tricks werden natürlich honoriert.

Logo goes Basic...

Wer einfache Logo-Grafik in Basic erzeugen möchte, dem hilft unser kleines Unterprogramm. Mit der Angabe von Winkel, Länge und Farbe wird eine entsprechende Linie von Startpunkt (0,0) aus gezogen.

```
SCREEN 1,320,200,5,1
WINDOW 1,,(0,0)-(300,186),0,1
```

```
REM Dieser Aufruf zieht eine Linie von (0,0)
REM zur Mitte des Bildschirms
```

```
a = 35
b = 170
c = 3
CALL LG (a,b,c)
```

```
SUB LG (winkel,lan,farbe)
  STATIC
  pi = 3.141593
  LOCATE 1.1
  ra = (winkel+Wges)
  *pi/180
  x1 = lan*COS(ra)+x
  y1 = lan*SIN(ra)+y
  LINE(x,y)-(x1,y1),farbe
  x = x1
  y = y1
  Wges = winkel+Wges
END SUB
```

Das Listing enthält bereits ein kleines Beispielprogramm. Es erzeugt eine Linie mit der Länge 170 im Winkel von 35 Grad. Der Winkel ist in der Variablen »a«, die Länge in »b« angegeben, »c« enthält die Farbe.

(Albert Treytl/M. Jobst/rs)

Startup mit Abfrage

Kopieren Sie die CLI-Befehle nur bei Bedarf in die RAM-Disk. Die CLI-Befehle in der RAM-Disk sparen Zeit und schonen Ihre Disketten — falls Sie mit dem CLI arbeiten. Das Kopieren benötigt jedoch Speicherplatz und verlängert die Boot-Zeit. Benötigen Sie das CLI nicht, sondern möchten mit der Workbench arbeiten, so ist die Installation der RAM-Disk in der »Startup-Sequence« unnötig. Für diesen Fall können Sie sich zwei Startdisketten erstellen: eine ausschließlich für die Arbeit mit dem CLI, die zweite für die Workbench. Das kostet eine Diskette. Es gibt jedoch eine pfiffige Methode: Bauen Sie eine Abfrage in der »Startup-Sequence« ein. Der Trick, der hierzu angewendet wird, ist die Verwendung der Befehle FAILAT und CD:

Sie müssen Ihre Startdiskette (im allgemeinen eine Kopie der Original-Workbench) vorher präparieren.

1. Erzeugen Sie ein neues Dateiverzeichnis mit dem Namen »n«:

```
MAKEDIR n
```

2. In diesem Verzeichnis erzeugen Sie mit dem Editor ein File mit dem Namen »dummy«:

```
ED n/dummy
```

Es spielt keine Rolle, was in diesem File steht. Schreiben Sie einen Text und verlassen den Editor mit <ESC> <x> <RETURN>.

3. Ergänzen Sie diese Zeilen in der »Startup-Sequence«:

```
ECHO "wollen Sie im CLI bleiben j/n ?"
FAILAT 25
CD >nil:?
IF NOT EXISTS dummy
  SKIP cli
ENDIF
LOADWB
ENDCLI >nil:
LAB cli
CD :
ECHO "wollen Sie RAMCLI aktivieren ?"
FAILAT 25
CD >nil: ?
IF EXISTS dummy
  SKIP nein
ENDIF
; Ein Beispiel um die RAM-Disk mit
; CLI-Befehlen zu aktivieren
MAKEDIR ram:c
COPY c: ram:c
```

```
PATH ram:c ADD
LAB nein
CD :
ECHO "Initialisierung
beendet"
```

Wenn Sie nun Ihre Work-Disk neu starten, fragt der Amiga, ob Sie mit dem CLI und mit den CLI-Befehlen in der RAM-Disk arbeiten möchten. Die Erklärung zum Programm:

Das Dummy-Gerät »NIL:« wird gewinnbringend genutzt. Durch Umlenken der Helpfunktion »?« auf dieses Gerät verfügt der Benutzer im CLI über eine elegante Eingabefunktion. Nach Ausgabe des Helptextes auf das nicht existierende Ausgabegerät können Sie einen Text eingeben.

Dieser stellt nun das aktuelle Directory dar (CD-Funktion).

Jetzt sucht der Computer in diesem Directory nach dem File »dummy«. Findet er es, war die Eingabe »n«, da nur im Directory »n« ein File »dummy« existiert. Mit Hilfe dieses Tricks läßt sich auch eine Paßwortkontrolle realisieren. Sie brauchen nur ein Directory zu generieren, dessen Name Ihrem Codewort entspricht. In diesem Verzeichnis muß wiederum das File »dummy« stehen. Wird ein falsches Schlüsselwort eingegeben, so findet der Amiga die Datei nicht und beginnt erneut mit der Abfrage.

Dies hilft auf jeden Fall, um die neugierige Schwiegermutter vom Amiga fernzuhalten — falls sie nicht weiß, daß man die »Startup-Sequence« mit <CTRL D> abbrechen kann.

(Martin Horn/rs)

Batchfiles beschleunigen

Unter einem Batchfile versteht man eine Textdatei, die CLI-Befehle enthält. Man kann in diese Datei eine Reihe solcher Befehle unterbringen, und mit »EXECUTE Name des Batchfiles« werden diese Kommandos automatisch der Reihe nach ausgeführt. Die CLI-Befehle sind Programme wie jedes andere Anwendungsprogramm auch — sie müssen also erst von der Diskette geladen werden (sofern sie sich nicht in der RAM-Disk befinden). Die Ladezeiten fallen besonders ärgerlich auf, wenn in einem Batchfile mehrere ECHO-Befehle (funktioniert ähnlich PRINT in Basic) nacheinander auftauchen, und deshalb jedesmal neu geladen werden müssen. Ähnlich verhält es sich mit anderen Kommandos

bei der täglichen Arbeit mit dem CLI (CD...DIR,CD...DIR und so weiter). Neben der Möglichkeit, die Befehle in die schnelle RAM-Disk zu legen, gibt es eine weitere, das RAM einzubeziehen. Man ordnet dem Laufwerk DF0: einen bestimmten RAM-Bereich als Puffer zu, in dem die zuletzt geladenen Befehle gespeichert werden. So brauchen diese Kommandos nicht neu geladen, sondern können aus dem Puffer gelesen werden. Um mit dem Befehl

ADDBUFFER Laufwerk Puffergröße

wirklichen Zeitgewinn erzielen zu können, sollte man den Puffer etwa 18 bis 20 KByte groß wählen. Beispiel: ADDBUFFERS df0: 18. Der einzige Nachteil dieser Methode soll jedoch nicht verschwiegen werden. Es ist nur durch einen Reset möglich, diesen Speicherbereich wieder anderweitig zu vergeben — man sollte sich also genau überlegen, wieviel Speicher man dafür »opfert«.

(Michael Baas/rs)

Gut gescrollt ist halb...

Die Funktion »ScrollRaster« aus der »graphics.library« ermöglicht dem (C-)Programmierer, einen rechteckigen Bildausschnitt in eine beliebige Richtung zu scrollen. Der Befehl hat die Syntax

```
ScrollRaster(rastport,deltaX,
deltaY,vonX,vonY,bisX,bisY);
```

»deltaX« und »deltaY« geben die Verschiebung in Punkten an. Hier beginnt auch schon das Problem: Normalerweise müßten positive Delta-Werte eine Verschiebung nach unten beziehungsweise rechts, negative Werte eine Verschiebung nach oben beziehungsweise links bewirken.

Daß dem nicht so ist, scheinen nicht einmal die Entwickler selbst bemerkt zu haben. Jedenfalls wird sowohl im »Amiga-Programmier-Handbuch« als auch im »ROM-Kernel-Manual« der Befehl mit oben genannten Scroll-Richtungen angegeben. Als Programmierer ist man spätestens dann verblüfft, wenn sich der Bildschirm, trotz scheinbar korrekter Syntax in die falsche Richtung davon macht. Erst ein Blick in Anhang A des »ROM-Kernel-Manual« offenbart die korrekte (ungewöhnliche) Syntax des Befehls. Zitat: »Die Funktion scrollt die Pixel auf den Punkt (0,0) zu.« Aha. Die korrekten Delta-Werte lauten also: nach oben beziehungsweise links: positiver Wert; nach unten beziehungsweise rechts: negativer Wert.

Achtet man bei der Anwendung dieses recht nützlichen Befehls auf diese »verdrehte« Form, bleibt so mancher Ärger erspart.

(Holger Schemel/M. Jobst/rs)

Bildschirmsplitting

Diese kleine Farbdemonstration splittet den Bildschirm in zwei Hälften. Das Programm demonstriert die Abfrage der Position des Rasterstrahls mit Hilfe der Hardware-Register.

```
execbase = 4
forbid = -132
permid = -138
start:
movem.l d0-d2/a1/a2/a6,-(SP)
lea $dff004,a1 ; u.a. Y-POS des Rasterstrahls
lea $dff180,a2 ; Farbregister
move.l execbase,a6
jsr forbid(a6) ; Multitasking aus
zeilenloop:
move.l #$2b00,d0 ; >>2b<< = Zeile für Split
; >>1b<< = Minimum
Farbloop:
move.w #$0000,d1 ;erste Farbe
wait:
move.l (a1),d2 ;
and.l #$0001ff00,d2; y-Pos. in d2
cmp.l d0,d2 ; Zeile erreicht ?
bne wait
move.w d1,(a2) ; neue Farbe ins Register
waitl:
```

```
move.l (a1),d2
and.l #$0001ff00,d2
cmp.l d0,d2
beq waitl ; nächste Zeile abwarten
cmp.b #$bc,$bfe001
beq done ;linke Maustaste gedrückt ?
add.w #$100,d1 ;neue Farbe
cmp.w #$1000,d1 ;Farbe wechselt laufend
blt wait
add.l #$0100,d0 ;Splitzeile +1
cmp.l #$13900,d0 ;größer Maximum?
blt farbloop
bge zeilenloop
done:
move.l execbase , a6
jsr permid(a6)
movem.l (sp)+,d0-d2/a1/a2/a6
rts
```

Das Programm ist mit dem Seka-Assembler geschrieben. Nach dem Start wird zunächst jeder weitere Task gesperrt. Danach verändert das Programm ab einer bestimmten Bildschirmzeile die Hintergrundfarbe. Zur besseren Demonstration wird diese Farbe laufend geändert. Nach Durchlaufen der Farbskala erhöht das Programm die Zeile, in der der Bildschirm gesplittet werden soll. Sie können den Programmablauf jederzeit durch einen Druck der linken Maustaste stoppen. Ein Nachteil des Programms ist, daß der Prozessor auf eine bestimmte Rasterposition wartet und nichts anderes erledigen kann. Noch besser wäre es daher, die Farbe des Hintergrunds mit Hilfe des Coppers ab einer bestimmten Position zu verändern.

(Klaus Kuphal/rs)

Animation in Basic

Auch von Basic aus ist es möglich, fließende Animationen zu erstellen. Der Trick besteht darin, mit zwei verschiedenen Screens zu arbeiten. Während der eine Screen das aktuelle Bild der Animation zeigt, wird auf dem anderen »versteckt« bereits das nächste erstellt. Anschließend werden beide Screens vertauscht, und der Vorgang beginnt von neuem. Das folgende kleine Demo-Programm arbeitet mit diesem Prinzip.

```
SCREEN 1,640,200,1,2 :REM zwei gleiche
SCREEN 2,640,200,1,2 :REM Screens öff
nen
WINDOW 1,"",(0,0)-(631,186),28,1 :REM zwei gleiche
WINDOW 2,"",(0,0)-(631,186),28,2 :REM Windows öffnen
```

Variablen:

```
x=10:y=10:x1=621:y1=176:w=1:p=1:f=1
```

Loop:

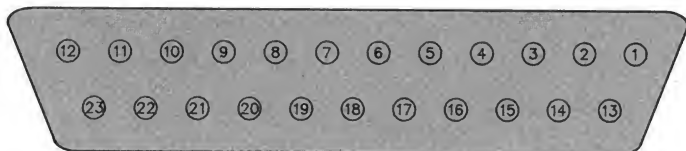
```
WINDOW w :REM Window
Nr. w nach vorne
w=w+1:IF w=3 THEN w=1
WINDOW OUTPUT w :REM Ausgabe
auf Hintergrundwindow
CLS :REM Window
löschen
LINE (x,y)-(x1,y1),1,b :REM Rechteck
zeichnen
x=x+f:y=y+p:x1=x1-f:y1=y1-p :REM Rechteck
verkleinern
IF y=170 THEN p=0
IF x=626 THEN f=-2:p=-1
IF y=0 THEN p=1
IF x=0 THEN f=2
GOTO Loop
```

Das Programm erzeugt eine einfache Animation eines sich in der Größe verändernden Rechtecks. (H. Kalläne/M. Jobst/rs)

Der Schalter für das Laufwerk

Einige Amiga-Programme funktionieren nur, wenn kein externes Laufwerk angeschlossen ist. Aber einige Laufwerke besitzen keinen Ausschalter. Damit Sie sich die Arbeit ersparen, das Laufwerk immer herauszuziehen, bauen Sie sich den Schalter selbst: Sie müssen ihn nur in die RDY-Leitung einlöten — entweder im Laufwerk oder im Anschlußkabel. Das RDY-Signal liegt beim Amiga 500 an Pin 1 des externen Diskanschlusses. Natürlich dürfen Sie den Schalter nur bei ausgeschaltetem Computer betätigen.

(Maurice AL-Khaliedy/rs)



SUB-D-Buchse		23pin/weiblich	
Pin	Signal	Beschreibung	
1	RDY	Bereitschafts-Signal	(active low)
2	DKRD	Schreib/Lese-Signal	(active low)
3	GND	Signalmasse	
4	GND	Signalmasse	
5	GND	Signalmasse	
6	GND	Signalmasse	
7	GND	Signalmasse	
8	MTRXD	Motor-Steuerung	(active low)
9	SEL2B	Laufwerk 2 aktivieren	(gepuffert, active low)
10	DRESB	Reset-Signal	(gepuffert, active low)
11	CHNG	Diskettenwechsel-Signal	(active low)
12	+5	+5 Volt	
13	SIDEB	Kopfwahl-Signal	(gepuffert, active low)
14	WPRO	Schreibschutz-Signal	(active low)
15	TK0	Spur 0-Signal	(active low)
16	DKWEB	Schreibfreigabe-Signal	(gepuffert, active low)
17	DKWDB	Daten schreiben	(gepuffert, active low)
18	STEPB	Signal für Kopf-Stepper	(gepuffert, active low)
19	DIRB	Stepper-Richtungs-Signal	(gepuffert, high = Richtung Spur 0)
20	NC	Nicht belegt	
21	SEL1B	Laufwerk 1 aktivieren	(gepuffert, active low)
22	INDEX	Index-Impuls	(active low)
23	+12V	+12 Volt	

Diese Tabelle zeigt die Belegung des 23poligen SUB-D-Steckers für die Zusatzlaufwerke am Amiga

Super-Hochauflösung

Überall steht geschrieben, der Amiga hätte eine maximale Bildschirmauflösung von 640 x 512 Punkten. In Wahrheit steht Ihnen eine weit höhere Auflösung zur Verfügung. Ein Screen kann bis zu 704 x 564 Punkte groß sein. Die 69376 zusätzlichen Punkte können Sie mit Hilfe der Grafikbefehle des Betriebssystems ohne weiteres nutzen. Besonders in C und Assembler ist dies recht einfach. Zwei Zahlen müssen geändert werden:

Um einen Screen zu öffnen, müssen Sie eine »NewScreen«-Struktur erstellen. Der dritte Wert dieser Tabelle gibt die Screen-Breite an, der vierte die Höhe. Üblicherweise stehen dort, beispielsweise für einen hochauflösenden Screen im Interlace-Modus die Werte 640 und 512. Ersetzen Sie diese Werte durch 704 beziehungsweise 564.

```
NewScreen.Width=704;  
NewScreen.Height=564;
```

Wenn Sie nun die Funktion »OpenScreen« anspringen, erscheint der übergroße Screen. Er ist aber nur zum Teil sichtbar. Um dem abzuweichen, müssen Sie folgendes ändern:

Erstens sollten Sie das Bild des Monitors etwas verkleinern und neu zentrieren. Die Regler hierzu befinden sich im allgemeinen an der Rückseite des Monitors.

Zweitens müssen Sie das Bild mit Hilfe der »Preferences« nach oben links verschieben. Klicken Sie hierzu das große Rechteck in

der Mitte des »Preference-Windows« an und bewegen das gesamte Fenster mit der Maus in die gewünschte Ecke. Für die genaue Einstellung ist ein wenig Experimentieren notwendig. Sie dürfen das Bild nicht zu weit verschieben, da sonst der obere Rand des Screens verschwindet. Wenn Sie den Monitor einmal eingestellt haben, können Sie in Ihren Programmen sofort die »Super-Hochauflösung« ausprobieren.

Der Effekt läßt sich im übrigen auch in allen anderen View-Modi ausnutzen, um mehr Punkte darzustellen. So stehen Ihnen zum Beispiel für einen LowRes-Screen 352 statt der üblichen 320 Punkte in der Horizontalen zur Verfügung.

Warum nicht gleich die volle Auflösung?

Die Einschränkung auf 640 x 512 Punkte erfolgt, weil nicht alle Monitore diese extreme Auflösung darstellen können (Over-scanning). Besonders an den Rändern kommt es zu Unregelmäßigkeiten. Außerdem sind in den Grenzbereichen nicht alle Sprites sichtbar.

(Jürgen Brendel/rs)

Beckertext und Bilder

Die Bildverarbeitung mit Beckertext ist bei der Verwendung des Hilfsprogramms »BTSnap« aufgrund des hohen Speicherbedarfs der Bilder stark eingeschränkt. Vor allem dann, wenn man nur 512 KByte Hauptspeicher zur Verfügung hat. Erstellt man jedoch seine Grafiken mit DPaint II und speichert die gewünschten Bildausschnitte als Pinsel (Brush), kann eine Menge Speicherplatz eingespart werden. Hinter dem Namen des Pinsels sollte man das Suffix ».IFF« setzen, damit das Bild von Beckertext erkannt wird.

(Jens Scheifgen/M. Jobst/rs)

Tolles Tool mit Anleitung

Der Editor »MicroEmacs« schlägt den ED um Längen. Allerdings nur wenige wissen, daß sich dieses hervorragende Werkzeug auf der ExtrasD-Diskette von Commodore befindet. Schauen Sie nach. Das Programm hat seinen Platz in der Schublade »Tools«. Und auch eine Anleitung ist auf der Diskette gespeichert. Sie verbirgt sich hinter dem Piktogramm »MicroEmacs.doc«. Es existieren drei Wege, die Anleitung zu lesen:

— Klicken Sie das Piktogramm »MicroEMACS.doc« zweimal an. Mit einem System-Requester fordert der Amiga Sie nun auf, das Original der ExtrasD-Diskette einzulegen. Sofern Sie nur ein Laufwerk besitzen, gelangen Sie nach zwei weiteren Diskettenwechseln in das erste Bild der Anleitung.

— Wer die Anleitung schneller lesen möchte, lädt das Utility »More«. Es dient dazu, ASCII-Dateien auf dem Bildschirm auszugeben. Nach der Frage »Filename?«, tippen Sie »MicroEMACS.doc« und drücken <RETURN>.

— Der elegante Weg ist, eine <Shift>-Taste zu drücken und mit der Maus die Piktogramme MicroEMACS.doc und More anzuklicken. Danach wählen Sie in der Menüzeile der Workbench »Open«. Wer es noch einfacher möchte, klickt bei gehaltener <Shift-Taste> MicroEMACS.doc an und aktiviert dann MORE mit einem Doppelklick.

Nach dem Laden sehen Sie auf dem Bildschirm die erste Seite der Anleitung. Durch Drücken von <h> gelangen Sie in einen Screen mit Hilfsanweisungen zu More. Mit <Space> blättern Sie in der Anleitung weiter. Sie umfaßt 70 Seiten und ist in englisch geschrieben. Am Ende befindet sich eine alphabetische Liste aller »Shortcuts« (abgekürzte Befehle) von MicroEMACS. Sie verlassen die Anleitung mit <q>. Wer den gesamten Text intensiv studieren möchte, kann ihn auch mit PrintFiles ausdrucken lassen.

(Nick Seggelke/rs)

Basic und Textverarbeitung?

Haben Sie sich nicht auch schon über den langsamen Editor des Amiga-Basic geärgert? Gerade wenn mit dem List-Fenster gearbeitet wird, dauert es in der Regel unzumutbar lange, bis der Text sich wieder der Größe des Fensters angepaßt hat.

Besser ist es daher, die Programme mit einem herkömmlichen Editor oder einer Textverarbeitung zu schreiben. Durch die intensive Nutzung der Funktionstastenbelegung oder Such- und Ersetzungsfunktionen der Textverarbeitungsprogramme läßt sich viel

Zeit einsparen. So können Sie beim Schreiben des Programms anstatt des Schlüsselwortes GOSUB die Folge »gb« einsetzen und diese vor dem Speichern durch GOSUB ersetzen. Ein Nachteil dieser Methode ist allerdings das Speichern und erneutes Laden in den Basic-Speicher vor der Ausführung des Programms. Folgendes ist jedoch zu beachten: Sowohl vom Basic-Interpreter als auch von den Textverarbeitungsprogrammen müssen die Programme im ASCII-Format gespeichert werden. Im Basic läßt sich dies durch die Angabe von »a« im Anschluß an den Dateinamen des SAVE-Befehls erreichen. (Lutz Beyert/rs)

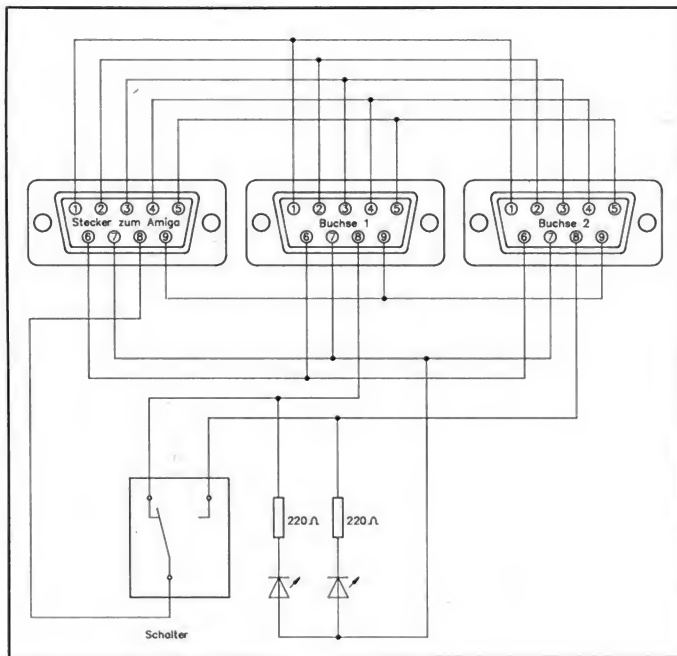
Maus rein — Maus raus

Das kann selbst ruhige Zeitgenossen manchmal zur Ver zweiflung bringen — immer wenn die Maus angeschlossen ist, braucht man einen Joystick und kurz darauf will ein anderes Programm nur mit der Maus arbeiten. Das damit verbundene häufige Umstecken der Stecker ist nicht nur lästig, es schadet auf lange Sicht auch den Kontakten am Joystick-Port des Amiga. Da liegt es nahe, sich einen Umschalter zu bauen. Hierzu benötigen Sie:

- einen 9-Pin-D-SUB-Stecker
- zwei 9-Pin-D-SUB-Buchsen
- zwei LEDs (rot 18 mA)
- zwei Widerstände 220 Ω
- einen Wechselschalter

Die Bauteile verbinden Sie nach dem im Bild gezeigten Schaltplan. Den fertigen Umschalter schließen Sie mit dem Stecker an den Port des Amiga an. In die Buchsen 1 und 2 stecken Sie den Joystick beziehungsweise die Maus ein. Über den Schalter können Sie nun einfach zwischen beiden hin- und herschalten. Die Leuchtdiode zeigt Ihnen jeweils an, wer aktiv ist.

(Maurice Al-Khaliedy/rs)



Der Schaltplan für den Maus/Joystick-Umschalter

Datum mit Format

Die Basic-Funktion »DATE\$« gibt das aktuelle Datum an. Leider erfolgt die Ausgabe nur im englischen Format. Anstatt »14.12.1988« heißt es »12-14-1988«. Mit folgendem Unterprogramm wird das anders:

Datum:

```
IF LEN(DATE$)=10 THEN
  Dat$=MID$(DATE$,4,2)+". "+LEFT$(DATE$,2)+". "+RIGHT$(DATE$,4)
END IF
```

```
IF LEN(DATE$)=9 AND LEFT$(DATE$,2)="-" THEN
  Dat$=MID$(DATE$,4,2)+". "+LEFT$(DATE$,1)+". "+RIGHT$(DATE$,4)
END IF
IF LEN(DATE$)=9 AND LEFT$(DATE$,2) "<" "-" THEN
  Dat$=MID$(DATE$,4,1)+". "+LEFT$(DATE$,2)+". "+RIGHT$(DATE$,4)
END IF
IF LEN(DATE$)=8 THEN
  Dat$=MID$(DATE$,3,1)+". "+LEFT$(DATE$,1)+". "+RIGHT$(DATE$,4)
END IF
RETURN
```

Nach dem Aufruf mit »GOSUB Datum« steht in der Variable »Dat\$« das aktuelle Datum im korrekten Format.

(Matthias Specht/M. Jobst/rs)

Selbstgestrickte Icons schützen

Mit Hilfe des CLI-Befehls PROTECT können Sie Ihre selbstgestellten Icons schützen. Dies ist besonders für Basic-Programmierer interessant.

Wenn Sie für Ihr Programm mühsam ein eigenes Icon erstellt haben, dann das Programm ändern und erneut speichern, löscht Amiga-Basic brutal Ihr Icon. Deshalb sollten Sie das Icon beziehungsweise die zugehörige Info-Datei mit diesem Befehl sichern:

```
PROTECT Programmtitel.info R
```

Jetzt hat Amiga-Basic keine Chance mehr. Dieses Icon kann nur noch gelesen werden.

(Rolf-Dieter Schiedrum/rs)

Unprotect

Eine geschützte Datei oder ein geschütztes Icon (siehe nebenstehender Tip) kann mit »DELETE« nicht mehr gelöscht werden. Dieser Schutz ist sinnvoll, um ein unbeabsichtigtes Löschen von Dateien zu vermeiden. Wenn Sie sich aber doch einmal fest entschlossen haben, eine gesicherte Datei zu verändern, können Sie den Schreibschutz aufheben:

```
PROTECT filename RWED
```

Dieser Befehl erlaubt Ihnen den freien Zugriff auf Ihre Datei.

(Laurent Münster/rs)

Druckermansteuerung in Basic

Die Ansteuerung eines Druckers von Basic aus ist etwas problematisch. Mit Hilfe von LPRINT, LPT1: und PRT: erfolgt die Druckeransteuerung über das »printer.device« des Amiga. Das Device versteht aber nicht die in den Druckerhandbüchern angegebenen Steuer codes, sondern benutzt eigene Kontrollzeichen. Steuert man dagegen einen Drucker, der an der parallelen Schnittstelle angeschlossen ist, direkt über PAR an, so sind die deutschen Umlaute nicht druckbar. Abhilfe schaffen die folgenden kurzen Unterprogramme »txt:« oder »stz:«, die den Drucker je nach Bedarf entweder über PRT oder direkt über PAR ansteuern. Da beim Amiga nur ein Druckerkanal auf einmal geöffnet sein darf, muß immer dafür gesorgt werden, daß jedes Unterprogramm den zuvor geöffneten Kanal schließt:

```
REM Demo für die Druckersteuerung
GOSUB txt
PRINT #1, "Dies ist ein Beispiel für die Umlaute:
äöüÄÖÜ"
GOSUB stz
PRINT #1, chr$(27); "W"; chr$(1); :REM Breitschrift
ein
GOSUB txt
PRINT #1, "Dies ist Breitschrift äöüÄÖÜ"
CLOSE #1
END
```

TIPS & TRICKS

stz: REM Steuerzeichen an Drucker
CLOSE #1 : OPEN "par:" for OUTPUT AS
#1 :RETURN
txt: REM Texte mit Umlauten
CLOSE #1 : OPEN "prt:" for OUTPUT AS #1
:RETURN
(Thomas Küpper/rs)

Scrollen im Listfenster

Das Listfenster des Amiga-Basic läßt sich auf mehrere Arten scrollen:

— Am bekanntesten ist das Scrollen mit Hilfe der Cursortasten.
— Seitenweise läßt sich das Fenster durch gleichzeitiges Drücken der Shift-Taste und der Cursortasten nach oben beziehungsweise unten verschieben.

— Die dritte Methode ist hingegen nicht so bekannt. Das Listfenster können Sie auch mit der Maus bewegen.

Um eine bestimmte Zeile auf dem Bildschirm anzuzeigen, brauchen Sie nur die linke Maustaste drücken und bei gedrückter Taste mit der Maus nach unten oder oben aus dem Fenster zu fahren. Das Listing bewegt sich so lange in die geforderte Richtung, bis Sie den Mauszeiger wieder ins Fenster bringen oder die Maustaste loslassen.

Diese Funktion dient eigentlich zum Markieren eines Blocks, der nachfolgend mit »COPY, CUT und PASTE« bearbeitet werden soll. Um den selektierten Block normal darzustellen, genügt ein Mausklick. Alles ist wie gehabt, nur der Cursor ist genau dort, wo Sie es möchten.
(Heinz Löschr/s)

Basic steuert die Hardware

In der AMIGA 8/9/87 steht in der Rubrik »Tips und Tricks«, wie sich die Power-LED mit Hilfe eines Assemblerprogramms ein- und ausschalten läßt. Auch Basicprogrammierer können auf die Hardware zugreifen:

POKE 12574721,254 schaltet die LED aus
POKE 12574721,252 schaltet die LED ein

Lassen Sie in Ihren Programmen die Leuchtdiode ein paarmal blinken. Verblüffen Sie mit diesem Effekt Ihre Bekannten.
(Daniel Swertz/rs)

RAM-Disk in neuem Kleid

Hatten Sie schon einmal das Bedürfnis, das Icon der RAM-Disk zu ändern, sind Sie daran aber gescheitert? Der folgende Trick hilft sicher: Soll das RAM-Disk-Icon aussehen wie das der Workbench, so fügen Sie einfach in der Startup-Sequenz vor der Zeile

PATH RAM:ADD

die Sequenz

COPY SYS:DISK.INFO RAM:

ein. Wollen Sie der RAM-Disk ein individuell gestaltetes Icon verpassen, so entwerfen Sie dieses zunächst wie jedes andere Disketten-Piktogramm mit dem Icon-Editor. Speichern Sie es dann unter dem Namen »DISK1.info« auf der Workbench-Diskette ab. Nun muß die an gleicher Stelle wie oben eingefügte Zeile lauten:

COPY SYS:DISK1.INFO RAM:DISK.INFO

Schon haben Sie beim nächsten Booten ein selbst gewähltes Disk-Icon für die RAM-Disk.
(Christoph Becker/M. Jobst/rs)

Die »Backups« von Ed und Edit

Wenn man viel mit dem Texteditor Ed oder dem Zeileneditor Edit arbeitet, kann es schon einmal passieren, daß man einen vorher in den Editor geladenen und veränderten Text mit <ESC> <x> überschrieben hat und später dringend auf die alte Version zurückgreifen möchte. Das ist vor allem dann der Fall, wenn man größere Textpassagen aus Versehen gelöscht und dies vor dem Speichern nicht bemerkt hat.

Weitgehend unbekannt ist jedoch, daß die Editoren vor dem Überschreiben eine Sicherheitskopie (Backup) der alten Datei im Unterverzeichnis »t« der Workbenchdiskette (V 33.56) anlegen. Diese Kopien heißen »ed-backup« beziehungsweise »editor-backup«. Mit der Anweisung »if "t/ed-backup"« können Sie eine solchermaßen gerettete Datei in den augenblicklich bearbeiteten Text einfügen.
(Robert Kretzschmar/rs)

Directory wird umgeleitet

Um mit dem CLI ein Directory zu Papier zu bringen, genügt die Umleitung der Ausgabe auf den Drucker mit »dir > prt:« oder »list > prt:«. Manchmal ist es jedoch von Vorteil, wenn Sie die Ausgabe erst einmal mit »dir > name« in eine Textdatei umleiten, um diese dann mit einem Editor oder dem Notepad zu verändern. So können Sie dem Dateinamen noch ergänzende Informationen, wie etwa Sinn und Zweck des Programms, wie man es gegebenenfalls installiert oder startet, hinzufügen und anschließend die Datei ausdrucken. Sie können diese Technik auch anwenden, wenn nur mit einem Teil der Dateien operiert werden soll: einfach den CLI-Befehl (hier: type) vor die Dateinamen der Textdatei schreiben und das Ganze als Batch-Datei ausführen.
(Lutz Beyert/rs)

Hilfe für den Floppyspeeder

»Facc« und »Facc II« sind nützliche Programme, die bei vielen Diskettenzugriffen Zeit sparen. Ein Nachteil ist, daß der Cache-Speicher — anders als beim Befehl ADDBUFFERS — von vornherein auf 256 KByte eingestellt ist. Wenn dies zuviel ist, der kann die Puffergröße zwar ändern, doch ist diese Arbeit vor jeder Benutzung des »Facc« lästig. Der folgende Patch schafft Abhilfe: — Facc: Laden Sie mit dem Filemonitor »NewZap« oder »FileZap« (Public Domain) den Sektor 14. An den Cursorpositionen 126 und 127 ist die Puffergröße gespeichert (\$0100 = 256). Ändern Sie den Wert nach Ihren Wünschen. — Facc II: Mit derselben Methode können Sie auch Facc II verändern, jedoch liegt die gesuchte Variable hier in Sektor 1 an der Position \$15C und \$15D. Arbeiten Sie auf jeden Fall mit einer Sicherheitskopie des Programms.
(Michael Holin/rs)

Echo mit Effekt

Wer Kommentare in Boot-Sequenzen etwas abwechslungsreicher gestalten will, dem kann nun geholfen werden. Es ist nämlich mit dem ECHO-Befehl des CLI möglich, verschiedene Schriftarten (kursiv, fett, unterstrichen etc.) auf Bildschirm (oder auch Drucker) auszugeben. Man muß dazu das Steuerzeichen »*e«, gefolgt von der Escape-Sequenz des gewünschten Modus, angeben. Die Escape-Sequenzen der einzelnen Modi finden Sie in Anhang C des Benutzerhandbuchs zu Ihrem Amiga. Dazu ein Beispiel:

ECHO "*e[4mDas*e[0m ist ja der *e[Wahnsinn*e[0m !!!"

bewirkt folgende Bildschirmausgabe:

Das ist ja der *Wahnsinn* !!!

(Christian Schmidt/M. Jobst/rs)

Texte drucken mit dem »Ed«

Vielleicht erstellen auch Sie Ihre Quellprogramme mit dem auf der Workbench mitgelieferten Texteditor »Ed«. In keiner Dokumentation findet man allerdings Informationen darüber, wie diese Texte auf den Drucker ausgegeben werden können. Natürlich gibt es eine umständliche Methode: mit »sa "name"« auf Diskette speichern und mit »type > prt: name« ausdrucken. Einfacher ist es, die im Speicher befindliche Datei gleich mit »sa "prt:"« auf den Drucker zu »speichern«. Mit dieser Technik können Sie nach vorheriger Markierung mit »bs« und »be« sowie »wb "prt:"« auch Zeilenbereiche auswählen und diese drucken. Haben Sie Ihren Drucker an der seriellen Schnittstelle angeschlossen, so ersetzen Sie die Gerätenamen »prt:« durch »ser:«.
(Edmund Olt/rs)

TIPS & TRICKS

Tips zu Superbase Professional

Die neuen Funktionen von Superbase Professional sind sehr leistungsfähig, wollen aber beherrscht sein. Wenn dazu das Handbuch kleine »Bugs« enthält, kann dies einem die Freude an der Arbeit schon verderben. Hier zeigen wir Ihnen, wie Sie Superbase optimal nutzen.

Einige der neuen Funktionen von Superbase Professional beziehungsweise Superbase 2 werden manchem Anwender — vor allem Einsteigern, die noch nie mit einer professionellen Datenbank gearbeitet haben — Probleme bereiten. Oftmals wird der Sinn und Zweck einer Funktion nicht verstanden, da zunächst nicht klar ist, wie man diese überhaupt einsetzen kann.

Eine dieser neuen Funktionen von Superbase ist die SER-Konstante. Im Handbuch können Sie nur lesen, daß diese SER-Konstante eine automatische Numerierung aller eingegebenen Datensätze bewirkt. Nehmen wir einmal an, Sie möchten mit Superbase Ihre Mitarbeiterdaten verwalten und alle Mitarbeiter mit einer Personalnummer versehen. Um dies zu erreichen, müssen Sie folgendermaßen vorgehen:

Die SER-Funktion

Als erstes erstellen wir eine neue Datei (Bild 1). Nachdem Sie den Namen der Datei — zum Beispiel »sbpeople« — eingegeben haben, erscheint das Ihnen sicherlich schon bekannte Kommunikationsfenster zur Dateidefinition. Jetzt definieren Sie als erstes ein numerisches Feld und geben ihm zum Beispiel den Namen »number«. Jetzt geben Sie zu diesem Feld folgende Konstante ein (Bild 2):

SER ("Dateiname")

Wenn Sie den oben genannten Namen für Ihre Datei genommen haben, müßte der



Bild 1. Die Definitionsmaske für eine Datei

Eintrag im Konstantenfeld jetzt folgendermaßen lauten:

SER ("sbpeople")

Im Superbase-Handbuch wurden leider die Anführungszeichen zwischen den Klammern vergessen. Sollten Sie es also nicht geschafft haben, eine SER-Konstante zu definieren, wissen Sie jetzt, wo der Fehler lag.

Automatische Numerierung

Nun müssen Sie nur noch die anderen Felder, die zu einer solchen Datenbank gehören, definieren. Dies könnten Felder sein wie Abteilung, Titel etc. Bei der Eingabe Ihrer Daten wird jetzt der erste Mitarbeiter die Nummer eins erhalten, der zweite die Nummer zwei etc.

Wenn Sie später einen neuen Mitarbeiter in Ihr Team aufnehmen wollen, müssen Sie nur einen neuen Datensatz öffnen, und schon hat Ihr neuer Mann seine Nummer. Sollte je-

mand ausscheiden, brauchen Sie nur den entsprechenden Datensatz zu löschen. Da die SER-Funktion als Konstante definiert ist, wird die Datei nicht neu durchnummeriert, sondern jeder Datensatz behält seine Nummer.

In diesem Zusammenhang ist es vielleicht noch erwähnenswert, daß eine Datei, bei der die SER-Funktion genutzt wurde, nicht reorganisiert werden kann. Sollten Sie es doch versuchen, erscheint eine Fehlermeldung wie in Bild 3 dargestellt.

Das Ganze mag Ihnen zunächst sehr unsinnig erscheinen, die Erklärung ist aber ganz einfach. Nehmen wir an, Sie hätten verschiedene Datensätze gelöscht und Superbase würde einen solchen Reorganisationslauf durchführen, dann wäre die ganze Datei neu durchnummeriert. Können

Reorganisation nicht möglich

Sie sich das Durcheinander vorstellen, wenn Mitarbeiter XY plötzlich die Personalnummer 12 besitzt und vor der Reorganisation war es die Nummer 4. Sollten Sie dennoch den



Bild 2. So geben Sie die SER-Funktion ein

Wunsch verspüren, Ihre Artikeldatei zu reorganisieren, ist dies nur über die Import/Export-Funktion möglich. Exportieren Sie zuerst alle Daten Ihrer Datei, mit Ausnahme des Feldes, auf die Sie die SER-Funktion angewandt haben. Danach erstellen Sie eine neue Datei nach dem Vorbild der alten und importieren die Daten wieder.

Achtung: Wenn Sie Schwierigkeiten mit der Import/Export-Funktion haben, empfehlen wir Ihnen, das entsprechende Kapitel im Handbuch noch einmal genau durchzulesen. Versuchen Sie nicht, nach dem Exportieren der Daten aus Ihrer Datei alle Datensätze zu löschen und dann die ASCII-Datei mit Ihren Daten wieder zu importieren. Die SER-Konstante einer Datei können Sie nachträglich nicht mehr löschen. Wenn in Ihrer Datei schon 70 Datensätze gespeichert waren und Sie importieren dieselbe ASCII-Datei, die Sie vorher schon exportiert haben, erhält der erste eingeleseene Datensatz die Nummer 71.

Mehrfacheintragungen

Ebenfalls eine neue Funktion ist die Möglichkeit der Mehrfacheintragungen in Textfeldern (Bild 4). In Textfeldern mit Mehrfacheintragungen wird immer nur der erste Eintrag angezeigt, alle anderen Einträge sehen Sie nur, wenn Sie die Tasten <CTRL P> oder <CTRL N> innerhalb dieses Textfeldes betätigen. Im Superbase-Handbuch ist ein schönes Beispiel abgedruckt, das Ihnen zeigt, wie man eine ganze Adresse in einem Feld unterbringt.

Toll, jetzt kann ich ja viel Platz sparen, werden Sie nun vielleicht denken. Früher brauchte man vier bis fünf Datenfelder für eine Adresse und ab sofort benötigt man nur noch ein Feld.

Spätestens wenn Sie versuchen, Ihre Adressen auf Etiketten auszudrucken oder einen Serienbrief zu schreiben, werden Sie feststellen müssen, daß dies mit Eintragungen aus Mehrfachtextfeldern nicht möglich ist. Die einzige Möglichkeit, die Mehrfacheintragungen auszuwerten, besteht mit der Filter-Funktion von Superbase. Wenn Sie zum Beispiel alle Meier in Ihrem zweiten Mehrfachtextfeld suchen,



Bild 3. Die Reorganisation einer mit der SER-Funktion bearbeiteten Datei ist nicht möglich

müssen Sie folgende Syntax verwenden:

FELDDNAME(1) LIKE "Meier"

Beachten Sie, daß Sie beim Anzeigen Ihrer Filterliste kein Feld mit dem Eintrag »Meier« sehen werden, da immer nur der erste Eintrag angezeigt wird. Leider ist auch in diesem Fall das im Handbuch angegebene Beispiel nicht ganz korrekt. Die Syntax FELDDNAME(2) bezieht sich nicht — wie im Handbuch erläutert — auf den zweiten Mehrfacheintrag, sondern auf den dritten. Dieses Beispiel sollte Ihnen zeigen, daß Sie die Mehrfacheintragungen in Textfeldern nur als zusätzliche Informationen verwenden sollten. Sicher waren Sie schon in der Situation, daß Sie reihenweise Daten eingeben mußten und es Ihnen nach einiger Zeit sicher auf die Nerven gegangen ist, daß Sie jeden eingegebenen Daten-

satz einzeln speichern müssen.

Mit den neuen Superbase-Versionen wird dieses Problem auch gelöst. Sie haben die Möglichkeit, im Datensatzmenü den sogenannten Stapelmodus einzuschalten. Sobald dieser aktiviert ist, speichert Superbase die eingegebenen Datensätze nicht sofort, sondern das Programm behält Ihre Daten im Speicher, bis Sie den Stapelmodus wieder ausschalten. Erst dann werden alle Daten auf Diskette gesichert.

Es wird gestapelt

Es ist möglich, daß Superbase auch während der Stapelmodus eingeschaltet ist, auf die Diskette zugreift. Das entspricht zwar nicht ganz den Angaben im Handbuch, aber Sie

brauchen deswegen keine Angst zu haben. Alle eingegebenen Daten werden korrekt gespeichert. Auch wenn der Stapelmodus eingeschaltet ist, müssen Sie nach jedem eingegebenen Datensatz die Tasten <Amiga-rechts S> und <Amiga-rechts N> betätigen oder im Menü Datensatz die Option »Speichern« und »Neu« aufrufen.

Bei der Eingabe von großen Datenmengen sollten Sie den Stapelmodus immer wieder einmal abschalten, um zu speichern. Sollte plötzlich Strom ausfallen oder Ihr Rechner es vorziehen, zu meditieren, gehen nur wenige Daten verloren.

Einsatz einer RAM-Disk

Wenn Sie zu den Glücklichen gehören, deren Amiga mit genügend RAM ausgestattet ist, können Sie natürlich Ihre Dateien in einer RAM-Disk bearbeiten. Dazu müssen Sie im File-Requester lediglich vor dem Dateinamen »RAM:« eingeben. Bei der Bearbeitung einer Datei in der RAM-Disk wird deutlich, wie schnell der Amiga mit Superbase wirklich ist. Sobald Sie Ihre RAM-Datei wieder schließen, fragt Superbase automatisch, ob die Datei wieder auf die Diskette zurückgeschrieben werden soll.

In loser Folge finden Sie in den nächsten Amiga-Sonderheften weitere Tips & Tricks zu Superbase. Das nächste Mal werden wir uns mit Lokup- und der Tenär-Funktion beschäftigen. (Dietmar Inäbnit/sk)

Bild 4. Im Feld »Eintragungen« lassen sich Mehrfacheintragungen festlegen



Schach der Langeweile!

Strategen aufgepaßt! Mit Shogun präsentieren wir Ihnen ein ganz besonderes Programm: ein altes japanisches Brettspiel, das dem Schach sehr ähnlich ist.

Es ist schwül, die Luft ist stickig. Kein Laut durchdringt die trügerische Stille. Das Gefecht ist an einem entscheidenden Punkt angelangt. Die Redakteure fixieren sich aus den Augenwinkeln. Wer macht den ersten Fehler? Plötzlich gellt ein markerschütternder Schrei. Verdammt, das war einer zuviel, denke ich und überlege mir bereits eine neue Angriffstaktik, um meinem Gegner endgültig den Garaus zu machen.

Solch fesselnde Szenen können auch Sie erleben. Sie benötigen lediglich einen Amiga, Shogun und einen Mitspieler als Opfer. Haben Sie das alles? Gut, dann kann es ja losgehen.

Shogun wird von zwei Spielern gespielt. Das Ziel des Spiels ist es den gegnerischen Shogun mattzusetzen, oder alle seine Truppen aufzureiben. Die Spielregeln sind schnell erklärt: Das Spielfeld besteht aus einem 8 x 8 Felder großen Bereich, in dem Sie Ihre Spielsteine bewegen können. Jeder der acht Steine ist mit einer Zahl von 1 bis 4 gekennzeichnet, die sich nach jedem Zug ändert. Die Steine sind anfangs oben und unten in den Grundreihen aufgestellt (Bild 1). Die Anfangsaufteilung ist jeweils verschieden, die Zahlen auf den Spielfiguren besitzen also anfangs immer verschiedene Werte.

Der Stein mit der gelben Markierung ist der Shogun. Diese, dem König beim Schach entsprechende Figur, kann lediglich die Zahlen 1 und 2 annehmen. Die auf den Steinen aktuell angezeigte Zahl muß gezogen werden, wobei man sich nur waagerecht und senkrecht, nie diagonal fortbewegen kann. Die Zugrichtung kann vorwärts, rückwärts oder seitwärts sein. Bei jedem Zug darf man einmal(!) einen Richtungswechsel im rechten Winkel vornehmen. Trifft man bei einem Zug genau auf ein Feld, das bereits von einem gegnerischen Stein besetzt ist, so wird der Gegner aus dem Spiel genommen. Liegt ein Stein, egal ob ein eigener oder einer des Gegners, im Weg, so darf dieser nicht aus dem Spiel genommen und auch nicht übersprungen werden.

Ein Shogun ist bedroht, wenn eine gegnerische Figur diesen beim nächsten Zug schlagen könnte. Ihr Amiga meldet dies mit einem Warnton und der Textausgabe »Ihr Shogun ist bedroht«. Der Shogun muß dann geschützt werden, indem

- er auf ein unbedrohtes Feld gesetzt wird,
- er den bedrohenden Stein schlägt, wenn dieser nicht von einer anderen gegnerischen Figur gedeckt ist,
- ein anderer Stein die gegnerische Figur schlägt, oder

— eine andere Figur so zwischen Angreifer und Shogun gesetzt wird, daß der Bedroher nicht mehr auf das Feld des Shogun zugreifen kann.

Sekt oder Selters - das Spielende

Es gibt zwei Wege, das Spiel zu beenden: Erstens, den gegnerischen Shogun mattzusetzen, daß heißt ihn so zu bedrohen, daß er sich nirgendwohin mehr retten kann. Und zweitens, alle Soldaten des Feindes, bis auf einen zu vernichten (Bild 2).

Die Spielregeln hören sich eventuell etwas kompliziert an, spätestens nach den ersten Spielen sind Sie mit den Regeln jedoch bestens vertraut. Dann werden Sie viele listige Spielzüge entdecken, die dem Gegner den Angstschweiß auf die Stirn treiben. Ein Hinweis noch: die Wertigkeit der Spielfiguren ist auf den gleichen Feldern in einem Spiel immer gleichbleibend. Das Schema, nach dem die Figuren ihre Wertigkeit verändern, möchten wir Ihnen an dieser Stelle nicht verraten. Vielleicht finden Sie ja die Logik, die dahintersteht, heraus (kleiner Tip: auch das Spielfeld verändert seine Lage in jedem Spiel und es gibt Kategorien von Spielsteinen).

Insgesamt gibt es drei verschiedene Spielsteinsorten. Jedem Stein ist eine der Bezeichnungen a, b, c, oder d in einer festgelegten Reihenfolge zugeteilt:

a	b	c	d	Spielfigur
1	1	2	2	Shogun
1	3	2	4	Typ 1
1	2	4	3	Typ 2

Ein Shogun kann also die Zahlen 1 oder 2 zeigen, die anderen Steine die Zahlen 1 bis 4. Jedem der 64 Spielfelder ist stets eine der entsprechenden Bezeichnungen »a« bis »d« zugeordnet.

Jetzt gibt's keine Hinweise mehr, sonst sind Sie am Ende Ihren Gegnern vielleicht zu hoch überlegen, und das Spiel wird für die Gegner frustrierend.

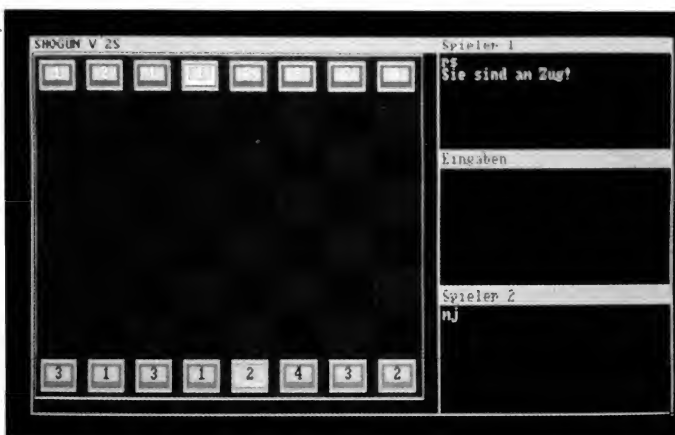


Bild 1. Die Anfangsszene von Shogun, noch sind alle Spielsteine auf dem Brett...



Bild 2. ...am Ende ist der Spieler Weiß geschlagen, da er neben dem Shogun nur noch eine weitere Figur besitzt.

Hinweise zum Programm

Geben Sie Listing 1 mit dem Checksummer (Seite 159) ein und speichern es unter dem Namen »Shogun« auf eine Diskette, auf der auch das Amiga-Basic enthalten ist. Darauf erstellen Sie ein Info-File mit demselben Namen und brauchen dieses nur noch zu laden.

Nach dem Starten des Programms erscheint das Titelbild. Sie können die Spielregeln noch einmal lesen, oder direkt ins Spiel verzweigen (<n> eingeben). In diesem Fall erscheint die Meldung »Einen Moment bitte...«, kurz darauf wird das Feld aufgebaut.

Auf diesem Bildschirm sind zusätzlich drei weitere Fenster zu sehen: Das Fenster »Eingaben« dient zur Eingabe von Daten bei Programmabfragen. Die Fenster »Spieler 1« und »Spieler 2« sind reine Ausgabefenster, hier erscheinen im Programmverlauf verschiedene Informationen.

Geben Sie nun im Eingabefenster die Namen für Spieler 1 und 2 an, indem Sie die Namen eintippen und dies mit <RETURN> bestätigen. Anschließend erscheint im Feld »Spieler 1« die Aufforderung »Sie sind am Zug«.

Dieser fährt nun mit dem Mauszeiger auf die Figur, die er bewegen möchte. Er klickt sie mit der linken Maustaste an, anschließend fährt er auf das Zielfeld und drückt erneut die linke Maustaste. Der Zug wird ausgeführt und der zweite Spieler ist an der Reihe.

War der angegebene Zug nicht regelgemäß, erscheint eine Fehlermeldung. Sie müssen sich dann ein anderes Zielfeld aussuchen. Entscheiden Sie sich nach dem Anklicken der gewünschten Figur doch für eine andere, klicken Sie die erste erneut an und dann die jetzt gewünschte.

Die Überprüfung, ob ein Shogun »matt« gesetzt ist, übernimmt das Programm. Dies kann bei seltenen, ungünstigen Situationen einige Sekunden dauern. Dies wird mit einer Meldung bekanntgegeben.

Ist das Spiel beendet, fragt Sie das Programm, ob eine weitere Partie gewünscht wird. Geben Sie daraufhin <j> oder <n> ein. Werden die Eingaben nicht angenommen, kann es daran liegen, daß das Eingabefenster im Moment nicht das aktuelle ist. Sie erkennen das an der »Geisterschrift«. Ein kurzer Klick auf die linke Maustaste genügt dann, der Mauszeiger muß sich dabei im Eingabefenster befinden.

Nun aber los: hinsetzen, abtippen, spielen.

(Frank Garnath/rs)

Programmname: Shogun

Computer: Amiga 500, 1000, 2000
mit Kickstart 1.2

Sprache: Amiga-Basic

Programm : Shogun

```

1 H00 ' *****
2 WD ' * Programm SHOGUN V 2S *
3 mV ' * von Frank Garnath 1987/1988 *
4 KR ' *****
5 zb3 CLEAR ,,30000
6 Af DEFINT f,m-s,w,z
7 TW SCREEN 2,640,215,3,2
8 fn WINDOW 7,"", (0,0)-(631,190),0,2
9 TV WINDOW 8,"", (170,72)-(473,127),0,2:LINE (0,0)-(303,65),1
    ,bf
10 ot WINDOW 6,"", (162,68)-(465,123),0,2
11 IN PALETTE 0,,.9,.9,1
12 JT FOR x=1 TO 7
13 Lp5 PALETTE x,x/10-.1,x/10-.1,x/10+.1
14 9e3 NEXT x
15 OI COLOR 1,0
16 lr LOCATE 1,12:PRINT" S H O G U N "
17 TN LOCATE 3,9:PRINT"von Frank Garnath"
18 lN LOCATE 5,9:PRINT"Lübeck, 1987/1988"
19 kP LOCATE 7,1:PRINT"Möchten Sie die Anleitung lesen (j/n)?"

20 yk farbe=1: add=1
21 Dx WHILE i$ <> "j" AND i$ <> "n"
```

```

22 PY5 i$=INKEY$:RANDOMIZE TIMER
23 MK WINDOW OUTPUT 7
24 Qp LINE (x,y)-(631-x,200-y),farbe,b:LINE (x+1,y)-(630-x,2
    00-y),farbe,b
25 dV farbe=farbe+add
26 yd IF farbe=7 THEN add=-1
27 lY IF farbe=1 THEN add=1
28 Lt IF z=0 AND x<631 THEN x=x+2:y=y+1 ELSE z=1:x=x-2:y=y-
    1:IF x<0 THEN z=0
29 ui3 WEND
30 28 CLS
31 l1 WINDOW CLOSE 6
32 Cy WINDOW CLOSE 8
33 lW IF i$="j" THEN CALL Anleitung
34 JA GOSUB Dunkel:PALETTE 2,0,0,0:COLOR 2,0
35 lB CLS:LOCATE 12,30:PRINT"Einen Moment bitte!"
36 qn GOSUB Hell
37 jz0 Voreinstellungen:
38 zN3 DIM SHARED f(63), wert(63), ZugR(7,15), ZugW(7,15)
39 WQ1 NeuesSpiel:
40 ok3 RANDOMIZE TIMER:v=INT(RND*4)
41 2K IF v=0 OR v=2 THEN RESTORE Daten0:f1=2:f2=3
42 vk IF v=1 OR v=3 THEN RESTORE Daten90:f1=3:f2=2
43 MU IF v>1 THEN a=1:b=0:c=-1 ELSE a=0:b=1:c=1
44 8s FOR y=a TO b STEP c
45 AD5 IF v>1 THEN d=15+y*32:e=y*32:f=-1 ELSE d=y*32:e=15+y*
    32:f=1
46 bY FOR x=d TO e STEP f
47 Et7 READ wert(x):wert(x+16)=wert(x)
48 hc5 NEXT x
49 kG3 NEXT y
50 Wn FOR y=0 TO 3
51 Gw5 FOR x=y*16 TO 6+y*16 STEP 2
52 Nk7 f(x)=f1:f(x+1)=f2:f(x+8)=f2:f(x+9)=f1
53 mH5 NEXT x
54 pL3 NEXT y
55 4D RANDOMIZE TIMER
56 l7 FOR x=0 TO 7
57 Nq5 prx(x)=1+x*48:pry(x)=1
58 OF pwx(x)=prx(x):pwy(x)=pry(x)+168
59 Jr zr(x)=wert(x):zw(x)=wert(x+56)
60 Ze v=RND
61 71 IF v<.5 THEN zar(x,1)=1:zar(x,2)=3:zar(x,3)=2:zar(x,4
    )=4
62 Em IF v>=.5 THEN zar(x,1)=1:zar(x,2)=2:zar(x,3)=4:zar(x,
    4)=3
63 ch v=RND
64 lW IF v<.5 THEN zaw(x,1)=1:zaw(x,2)=3:zaw(x,3)=2:zaw(x,4
    )=4
65 jC IF v>=.5 THEN zaw(x,1)=1:zaw(x,2)=2:zaw(x,3)=4:zaw(x,
    4)=3
66 zU3 NEXT x
67 XI zar(3,1)=1:zar(3,2)=1:zar(3,3)=2:zar(3,4)=2
68 Ap zaw(4,1)=1:zaw(4,2)=1:zaw(4,3)=2:zaw(4,4)=2
69 M3 IF ns=1 THEN Spielfeld
70 9n FOR y=.75 TO .02 STEP -.01
71 fy5 FOR x=0 TO 1
72 TV7 PALETTE x,y,y+.22,y+.25
73 6b5 NEXT x
74 9f3 NEXT y
75 7Y FOR x=2 TO 7
76 bF5 PALETTE x,.02,.24,.27
77 Af3 NEXT x
78 rc WINDOW CLOSE 7
79 5u WINDOW 2,"SHOGUN V 2S", (0,0)-(397,200),0,2
80 9Z FOR x=0 TO 8
81 oS5 LINE (10,5)-(387+x,194+INT(x/2)),2,b
82 PU3 NEXT
83 zNO Spielfeld:
84 VL3 WINDOW OUTPUT 2:LOCATE 2,1
85 QS LINE (0,0)-(385,193),5,b
86 Et FOR y=1 TO 163 STEP 48
87 J65 FOR x=1 TO 317 STEP 96
88 KM7 LINE (x,y)-(x+47,y+23),f1,bf
89 Px LINE (x+48,y)-(x+95,y+23),f2,bf
90 wU LINE (x,y+24)-(x+47,y+47),f2,bf
91 KN LINE (x+48,y+24)-(x+95,y+47),f1,bf
92 Pu5 NEXT x
93 Sy3 NEXT y
94 K80 Figuren:
95 Lk3 FOR x=0 TO 7
96 Ip5 LINE (6+x*48,4)-(41+x*48,21),4,bf
97 Wt LINE (6+x*48,172)-(41+x*48,189),1,bf
98 Cl LINE (8+x*48,6)-(39+x*48,19),6,b
```



```

99 oX LINE (8+x*48,174)-(39+x*48,187),7,b
100 of IF x<>3 THEN PAINT (20+x*48,12),6
101 wu IF x<>4 THEN PAINT (20+x*48,179),7
102 Z43 NEXT x
103 iU PAINT (164,12),5,6:PAINT (212,179),5,7
104 xN COLOR 5,4
105 Vu FOR x=0 TO 7
106 Pr5 PRINT PTAB(12+x*48)zar(x,zr(x));
107 e93 NEXT x
108 Ow COLOR 2,1:LOCATE 23,1
109 Zy FOR x=0 TO 7
110 xZ5 PRINT PTAB(12+x*48)zaw(x,zw(x));
111 iD3 NEXT x
112 B10 Eingaben:
113 bu3 WINDOW 4,"Spieler 1",(404,0)-(631,54),0,2
114 jL WINDOW 5,"Spieler 2",(404,143)-(631,200),0,2
115 Xe WINDOW 3,"Eingaben",(404,67)-(631,129),0,2
116 Rv PALETTE 0,.02,.24,.27
117 j1 PALETTE 1,1,1,1
118 d1 PALETTE 2,0,0,0
119 tM PALETTE 3,.33,.33,.33
120 an PALETTE 4,.93,.2,0
121 iA PALETTE 5,1,1,.13
122 iW PALETTE 6,.53,.15,0
123 oF PALETTE 7,.63,.63,.63
124 xJ COLOR 5,0
125 dy FOR x=1 TO 2
126 D65 PRINT "Name Spieler",x:INPUT na$(x)
127 C1 BEEP
128 zU3 NEXT x
129 oO WINDOW OUTPUT 4:COLOR 4,0:PRINT na$(1)
130 ys WINDOW OUTPUT 5:COLOR 1,0:PRINT na$(2)
131 ro WINDOW OUTPUT 3:COLOR 5,0
132 t5 CLS:sp=1
133 eR0 naechsterSpieler:
134 yk3 IF sp=0 THEN sp=1 ELSE sp=0
135 tu SteinNr=-2
136 bF0 Spielbeginn:
137 Mc3 IF mattpr=1 THEN Mattpruef
138 VO WINDOW OUTPUT 4+sp:COLOR 4-sp*3,0:CLS:PRINT na$(1+sp)
139 bu PRINT "Sie sind am Zug!":PRINT p$
140 i3 WINDOW OUTPUT 2
141 wy1 CheckMouse1:
142 nx3 WHILE MOUSE(0)<1
143 kY WEND
144 J3 mx=MOUSE(3):my=MOUSE(4)
145 OQ mx=INT(mx/48)*48+1:my=INT(my/24)*24+1
146 d3 IF sx=0 THEN sx=1:sy=1
147 vC IF px=0 THEN px=1:py=1
148 2T LINE (sx,sy)-(sx+47,sy+23),f(sx/48+sy/3),b
149 q1 LINE (px,py)-(px+47,py+23),f(px/48+py/3),b
150 Ed FOR x=0 TO 7
151 S15 IF prx(x)=mx AND pry(x)=my THEN m=x:GOTO MarkSetz
152 Ns3 NEXT x
153 Y9 GOTO CheckMouse1
154 Yu1 MarkSetz:
155 tS3 BEEP:LINE (mx,my)-(mx+47,my+23),5,b
156 DG1 CheckMouse2:
157 2K3 WINDOW OUTPUT 2
158 3D WHILE MOUSE(0)<1
159 Oo WEND
160 95 sx=MOUSE(3):sy=MOUSE(4)
161 6Z IF sx>383 OR sy>191 THEN CheckMouse2
162 j3 sx=INT(sx/48)*48+1:sy=INT(sy/24)*24+1
163 rk IF sx=prx(m) AND sy=pry(m) THEN LINE (sx,sy)-(sx+47,sy+23),f(sx/48+sy/3),b:GOTO Spielbeginn
164 iA LINE (sx,sy)-(sx+47,sy+23),5,b
165 Oa1 EinsprungMattpruef:
166 713 schrittzahl=ABS(INT((prx(m)-sx)/48))+ABS(INT((pry(m)-sy)/24))
167 Dp IF schrittzahl<>zar(m,zr(m)) THEN
168 9J5 fz=1:fz1$="Sie müssen":fz2$="Feld(er) ziehen!"
169 O5 GOTO FalscherZug
170 gZ3 END IF
171 Zy FOR x=0 TO 7
172 OP5 IF sx=prx(x) AND sy=pry(x) THEN
173 C77 IF schwpr=1 THEN Sprungpruef
174 Wf fz1$="Sie können nicht Ihren":fz2$="eigenen Stein so hlagen!"
175 6B GOTO FalscherZug
176 mf5 END IF
177 mH3 NEXT x
178 aa1 Sprungpruef:
179 YJ3 IF zar(m,zr(m))=1 THEN AllgShogunpruef1

```

```

180 eL weg1=0:weg2=0
181 Ze IF sx=prx(m) THEN ZugpruefY
182 iJ IF sy=pry(m) THEN ZugpruefX
183 8t IF sx<prx(m) THEN stpx=48
184 2M IF sx>prx(m) THEN stpx=-48
185 qn IF sy<pry(m) THEN stpy=24
186 yK IF sy>pry(m) THEN stpy=-24
187 i2 FOR n=sx+stpx TO prx(m) STEP stpx
188 yx5 FOR o=0 TO 7
189 wd7 IF (prx(o)=n AND pry(o)=sy) OR (pwx(o)=n AND pwy(o)=sy) THEN weg1=1
190 te IF (prx(o)=n-stpx AND pry(o)=pry(m)) OR (pwx(o)=n-stpx AND pwy(o)=pry(m)) THEN weg2=1
191 i45 NEXT o
192 h23 NEXT n
193 5k FOR n=sy+stpy TO pry(m)-stpy STEP stpy
194 435 FOR o=0 TO 7
195 007 IF (pry(o)=n AND prx(o)=prx(m)) OR (pwy(o)=n AND pwx(o)=prx(m)) THEN weg1=1
196 iC IF (pry(o)=n AND prx(o)=sx) OR (pwy(o)=n AND pwx(o)=sx) THEN weg2=1
197 oA5 NEXT o
198 n83 NEXT n
199 66 IF weg1=1 AND weg2=1 THEN FZug
200 Gr GOTO AllgShogunpruef1
201 p71 ZugpruefX:
202 OH3 IF sx<prx(m) THEN stp=48
203 dD IF sx>prx(m) THEN stp=-48
204 7f FOR n=sx+stp TO prx(m)-stp STEP stp
205 FE5 FOR o=0 TO 7
206 JS7 IF (prx(o)=n AND pry(o)=sy) OR (pwx(o)=n AND pwy(o)=sy) THEN FZug
207 yK5 NEXT o
208 xI3 NEXT n
209 PO GOTO AllgShogunpruef1
210 3M1 ZugpruefY:
211 sO3 IF sy<pry(m) THEN stp=24
212 PA IF sy>pry(m) THEN stp=-24
213 Pz FOR n=sy+stp TO pry(m)-stp STEP stp
214 ON5 FOR o=0 TO 7
215 Rn7 IF (pry(o)=n AND prx(o)=sx) OR (pwy(o)=n AND pwx(o)=sx) THEN
216 Fd1 FZug:
217 O59 fz1$="Sie dürfen keinen Stein":fz2$="Überspringen!":fz=0
218 ns GOTO FalscherZug
219 TM7 END IF
220 BX5 NEXT o
221 AV3 NEXT n
222 CEO AllgShogunpruef1:
223 be3 spp=sp:sxx=sx:syy=sy
224 um zz=zr(m):zr(m)=wert(INT(sx/48+sy/3))
225 wM pxm=prx(m):pym=pry(m):prx(m)=sx:pry(m)=sy
226 AZ z=-1
227 JMO AllgShogunpruef2:
228 TF3 IF sp=0 THEN
229 2q5 sx=prx(3):sy=pry(3)
230 Wv FOR x=0 TO 7
231 hX7 IF spp=0 AND pwx(x)=sxx AND pwy(x)=syy THEN pxmem=pwx(x):pwx(x)=0:pymem=pwy(x):pwy(x)=0:z=x:pm=1
232 fA5 NEXT x
233 Fo GOSUB ShogunBedroht
234 Zv IF spp=0 THEN sp=1:GOSUB WerteTauschen:GOTO AllgShogunpruef2 ELSE GOTO Pruefende
235 O73 ELSE
236 H75 sx=prx(4):sy=pry(4)
237 d2 FOR x=0 TO 7
238 sj7 IF spp=1 AND pwx(x)=sxx AND pwy(x)=syy THEN pxmem=pwx(x):pwx(x)=0:pymem=pwy(x):pwy(x)=0:z=x:pm=1
239 mH5 NEXT x
240 Mv GOSUB ShogunBedroht
241 i1 IF spp=1 THEN sp=0:GOSUB WerteTauschen:GOTO AllgShogunpruef2
242 qJ3 END IF
243 jh1 Pruefende:
244 mu3 sx=sxx:sy=syy:sp=spp
245 Po GOSUB WerteTauschen
246 mn0 Steinsetz:
247 Tf3 IF spm=1 AND pm=1 THEN pm=0:IF z<>-1 THEN pwx(z)=pxmem:pwy(z)=pymem
248 fM IF spm=1 THEN MattEnde

```

Listing 1. »Shogun« bitte mit dem Checksummer (Seite 159) eingeben

```

249 pF WINDOW OUTPUT 4+sp:CLS:PRINT na$(1+sp)
250 Xp WINDOW OUTPUT 2
251 C1 BEEP
252 Te LINE (prx(3+sp),pry(3+sp))-(prx(3+sp)+47,pry(3+sp)+23),f
      (prx(3+sp)/48+pry(3+sp)/3),b
253 pG LINE (mx,my)-(mx+47,my+23),f(mx/48+my/3),bf
254 3D prx(m)=sx:pry(m)=sy:zr(m)=wert(INT(prx(m)/48+pry(m)/3))
255 12 LINE (prx(m)+5,pry(m)+3)-(prx(m)+40,pry(m)+20),4-sp*3,bf

256 H2 LINE (prx(m)+8,pry(m)+5)-(prx(m)+38,pry(m)+18),6+sp,b
257 DQ IF m=3+sp THEN fa=5 ELSE fa=6+sp
258 To PAINT (prx(m)+20,pry(m)+12),fa,6+sp
259 d1 LINE (prx(m),pry(m))-(prx(m)+47,pry(m)+23),f(prx(m)/48+p
      ry(m)/3),b
260 Br LOCATE pry(m)/8+2,1:COLOR 5-sp*3,4-sp*3
261 x6 PRINT PTAB(prx(m)+11);zar(m),zr(m))
262 g4 IF sp=5b THEN p$=""
263 op IF z<>-1 THEN GOSUB FigurGeschlagen
264 17 GOSUB WerteTauschen
265 CY GOTO naechsterSpieler
266 ga0 Summen:
267 ba3 zzz=zahl-1
268 Gy sumX(1,1)=0:sumX(1,2)=0:sumX(1,3)=-zahl*48:sumX(1,4)=zah
      l*48:sumY(1,1)=-zahl*24:sumY(1,2)=zahl*24:sumY(1,3)=0:su
      mY(1,4)=0
269 nY IF Pruefstufe=1 THEN RETURN
270 Po sumX(2,1)=-48:sumX(2,2)=48:sumX(2,3)=-48:sumX(2,4)=48:su
      mY(2,1)=-zzz*24:sumY(2,2)=zzz*24:sumY(2,3)=zzz*24:sumY(2
      ,4)=-zzz*24
271 vc IF Pruefstufe=2 THEN RETURN
272 GC sumX(3,1)=-zzz*48:sumX(3,2)=zzz*48:sumX(3,3)=-zzz*48:sum
      X(3,4)=zzz*48:sumY(3,1)=-24:sumY(3,2)=24:sumY(3,3)=24:su
      mY(3,4)=-24
273 3g IF Pruefstufe=3 THEN RETURN
274 Ud sumX(4,1)=-96:sumX(4,2)=96:sumX(4,3)=-96:sumX(4,4)=96:su
      mY(4,1)=-48:sumY(4,2)=48:sumY(4,3)=48:sumY(4,4)=-48
275 nP RETURN
276 Mw0 Mattpruef:
277 z63 SteinNr=SteinNr+1:m=SteinNr
278 13 IF SteinNr=-1 THEN IF sp=0 THEN m=3 ELSE m=4
279 2z IF SteinNr=0 THEN WINDOW OUTPUT 4+sp:PRINT"Einen Augenbl
      ick..." :WINDOW OUTPUT 2
280 B1 IF (sp=0 AND SteinNr=3) OR (sp=1 AND SteinNr=4) THEN Mat
      tpruef
281 Ef IF SteinNr>7 THEN VerlorenGewonnen
282 vJ IF prx(m)=0 THEN Mattpruef
283 zg Pruefstufe=1:pr=0
284 Vv zahl=zar(m),zr(m))
285 PB mx=prx(m):my=pry(m):zrzahl=zr(m)
286 7u1 EinsprungMEnde:
287 413 spm=1:MEnde=0
288 eM GOSUB Summen
289 ko1 Pruefstufen:
290 fC3 pr=pr+1
291 dB sx=prx(m)+sumX(Pruefstufe,pr):sy=pry(m)+sumY(Pruefstufe,
      pr):IF sx<1 OR sy<1 OR sx>337 OR sy>169 THEN MEnde=1
      :GOTO MattEnde
292 Rq IF pr>4 AND zahl=Pruefstufe AND SteinNr=7 THEN Verloren
      Gewonnen
293 cu IF pr>4 AND zahl=Pruefstufe THEN naechstStein=1:GOSUB M
      attEnde:GOTO Mattpruef
294 77 IF pr>4 AND zahl>Pruefstufe THEN Pruefstufe=Pruefstufe
      +1:MEnde=1:pr=0:GOTO MattEnde
295 rt IF Pruefstufe=5 THEN naechstStein=1:MEnde=0:GOSUB MattEn
      de:GOTO Mattpruef
296 5q GOTO EinsprungMattpruef
297 WV1 MattEnde:
298 xX3 spm=0:sx=0:sy=0:prx(m)=mx:pry(m)=my:zr(m)=zrzahl
299 QY IF MEnde=1 THEN EinsprungMEnde
300 ec IF naechstStein=1 THEN naechstStein=0:RETURN
301 Nc mattpr=0:SteinNr=-2
302 hu GOTO Spielbeginn
303 eV0 ShogunBedroht:
304 c53 FOR w=0 TO 7
305 175 shfeld=ABS((pwx(w)-sx)/48)+ABS((pwy(w)-sy)/24)
306 uR IF shfeld=zaw(w,zw(w)) THEN
307 q67 px=pwx(w):py=pwy(w)
308 4L IF px<>0 AND shfeld=1 THEN FZug2
309 9A IF px<>0 THEN GOSUB Shogunpruef
310 wp5 END IF
311 uO3 NEXT w
312 O0 RETURN
313 xD0 Shogunpruef:

```

```

314 oV3 weg1=0:weg2=0
315 IA IF sx=px THEN ShogunpruefY
316 RF IF sy=py THEN ShogunpruefX
317 MM IF sx<px THEN stpx=48
318 nb IF sx>px THEN stpx=-48
319 IK IF sy<py THEN stpy=24
320 nT IF sy>py THEN stpy=-24
321 UI FOR n=sx+stpx TO px STEP stpx
322 875 FOR o=0 TO 7
323 vH7 IF (sp=0 AND o=3) OR (sp=1 AND o=4) THEN
324 se9 IF pwx(o)=n AND pwy(o)=sy THEN weg1=1
325 5u IF pwx(o)=n-stpx AND pwy(o)=py THEN weg2=1
326 ra7 ELSE
327 Ar9 IF (prx(o)=n AND pry(o)=sy) OR (pwx(o)=n AND pwy(o)
      )=sy THEN weg1=1
328 7F IF (prx(o)=n-stpx AND pry(o)=py) OR (pwx(o)=n-stpx
      AND pwy(o)=py) THEN weg2=1
329 F87 END IF
330 xJ5 NEXT o
331 wH3 NEXT n
332 Ik FOR n=sy+stpy TO py-stpy STEP stpy
333 JI5 FOR o=0 TO 7
334 6S7 IF (sp=0 AND o=3) OR (sp=1 AND o=4) THEN
335 fS9 IF pwy(o)=n AND pwx(o)=px THEN weg1=1
336 1n IF pwy(o)=n AND pwx(o)=sx THEN weg2=1
337 217 ELSE
338 jX9 IF (pry(o)=n AND prx(o)=px) OR (pwy(o)=n AND pwx(o)
      )=px THEN weg1=1
339 Kv IF (pry(o)=n AND prx(o)=sx) OR (pwy(o)=n AND pwx(o)
      )=sx THEN weg2=1
340 QJ7 END IF
341 8U5 NEXT o
342 7S3 NEXT n
343 57 IF weg1=1 AND weg2=1 THEN RETURN
344 D6 GOTO FZug2
345 b11 ShogunpruefX:
346 uA3 IF sx<px THEN stp=48
347 od IF sx>px THEN stp=-48
348 2w FOR n=sx+stp TO px-stp STEP stp
349 ZY5 FOR o=0 TO 7
350 M17 IF (sp=0 AND o=3) OR (sp=1 AND o=4) THEN
351 UK9 IF pwx(o)=n AND pwy(o)=sy THEN RETURN
352 HO7 ELSE
353 lW9 IF (prx(o)=n AND pry(o)=sy) OR (pwx(o)=n AND pwy(o)
      )=sy THEN RETURN
354 eX7 END IF
355 M15 NEXT o
356 Lg3 NEXT n
357 QJ GOTO FZug2
358 r21 ShogunpruefY:
359 JA3 IF sy<py THEN stp=24
360 rh IF sy>py THEN stp=-24
361 NJ FOR n=sy+stp TO py-stp STEP stp
362 m15 FOR o=0 TO 7
363 Zv7 IF (sp=0 AND o=3) OR (sp=1 AND o=4) THEN
364 YS9 IF pwy(o)=n AND pwx(o)=sx THEN RETURN
365 UD7 ELSE
366 qZ9 IF (pry(o)=n AND prx(o)=sx) OR (pwy(o)=n AND pwx(o)
      )=sx THEN RETURN
367 rk7 END IF
368 Zv5 NEXT o
369 Yt3 NEXT n
370 n71 FZug2:
371 DM3 li=1
372 RF IF spp<>sp THEN sb=sp:p$="Ihr Shogun wird bedroht!":GO
      SUB ShogunWirdBedroht:RETURN
373 uT IF spp=sp THEN fz1$="Durch diesen Zug würde Ihr":fz2$="S
      hogun bedroht werden!":prx(m)=pxm:pry(m)=pym:zr(m)=zz:fz
      =0
374 oD IF z<>-1 THEN pwx(z)=pxmem:pwy(z)=pymem:pm=0
375 E4 sx=sxx:sy=syy
376 vR0 FalscherZug:
377 KK3 IF mattpr=1 THEN MEnde=1:GOTO MattEnde
378 UU IF li=1 AND ((sp=0 AND m=3) OR (sp=1 AND m=4)) THEN LINE
      (sx,sy)-(sx+47,sy+23),1+sp*3,b
379 dB IF li=1 AND ((sp=0 AND m<>3) OR (sp=1 AND m<>4)) THE
      N LINE (prx(3+sp),pry(3+sp))-(prx(3+sp)+47,pry(3+sp)+23)
      ,1+sp*3,b
380 u5 IF li=1 THEN LINE (px,py)-(px+47,py+23),1+sp*3,b
381 kP WINDOW OUTPUT 4+sp:CLS
382 Th PRINT fz1$
383 1W IF fz=1 THEN PRINT zar(m),zr(m));:fz=0
384 ap PRINT fz2$

```



```

385 10 WINDOW OUTPUT 2
386 ky SOUND 150,5,255,1
387 tK LINE (sx,sy)-(sx+47,sy+23),f(sx/48+sy/3),b
388 hZ LINE (px,py)-(px+47,py+23),f(px/48+py/3),b
389 HE IF 11=1 AND ((sp=0 AND m<>3) OR (sp=1 AND m<>4)) THE
N LINE (prx(3+sp),pry(3+sp))-(prx(3+sp)+47,pry(3+sp)+23)
,f(prx(3+sp)/48+pry(3+sp)/3),b
390 RZ li=0
391 Q2 GOTO CheckMouse2
392 jp0 ShogunWirdBedroht:
393 xe3 LINE (sx,sy)-(sx+47,sy+23),1+sp*3,b:LINE (px,py)-(px+47,
py+23),1+sp*3,b
394 Ps FOR x=230 TO 630 STEP 15
395 8f5 SOUND x,.5,255,1:SOUND 830-x,.5,255,2
396 Jo3 NEXT x
397 71 mattpr=1
398 m0 RETURN
399 170 FigurGeschlagen:
400 Gf3 FOR x=0 TO 7
401 qH5 IF pwx(x)<>0 THEN Anzahl=Anzahl+1
402 Pu3 NEXT x
403 8t IF sp=1 AND Anzahl<3 THEN sp=0:mattpr=0:GOTO VerlorenGe
wonnen
404 Cs IF sp=0 AND Anzahl<3 THEN sp=1:mattpr=0:GOTO VerlorenGe
wonnen
405 Hs Anzahl=0
406 uW RETURN
407 h90 WerteTauschen:
408 On3 FOR x=0 TO 7
409 nw5 SWAP prx(x),pwx(x):SWAP pry(x),pyw(x):SWAP zr(x),zw(x)
410 Qj FOR y=1 TO 4
411 m27 SWAP zar(x,y),zaw(x,y)
412 b75 NEXT y
413 a53 NEXT x
414 2e RETURN
415 R30 Dunkel:
416 dc3 FOR x=1 TO 0 STEP -.01
417 OD5 PALETTE 0,x,x,x:PALETTE 1,x,x,x
418 fA3 NEXT x
419 7j RETURN
420 xU0 Hell:
421 Co3 FOR x=0 TO 1 STEP .01
422 Ti5 PALETTE 0,x,x,x:PALETTE 1,x,x,x
423 kF3 NEXT x
424 Co RETURN
425 Fy0 VerlorenGewonnen:
426 T83 WINDOW OUTPUT 4+sp:CLS
427 q5 PRINT na$(1+sp),"
428 70 IF mattpr=1 THEN
429 gw5 mattpr=0:spm=0:p$=""
430 xL PRINT "Ihr Shogun ist matt!!!"
431 CU PRINT "(Er würde beim nächsten Zug"
432 JO PRINT "geschlagen werden.)"
433 QG PRINT "Sie haben leider verloren."
434 bK3 ELSE
435 Os5 PRINT "Sie haben nur noch zwei"
436 2C PRINT "Steine und somit verloren!"
437 11 PRINT "Üben Sie noch ein wenig!"
438 Ot3 END IF
439 qT WINDOW OUTPUT 5-sp:CLS
440 BT PRINT na$(2-sp),"
441 rP PRINT "Sie haben gewonnen!"
442 n1 PRINT "Herzlichen Glückwunsch!!!"
443 ky WINDOW OUTPUT 3
444 wD PRINT "Noch ein Spiel (J/N)?":i$="":p$=""
445 3n WHILE i$<>"j" AND i$<>"n"
446 YE5 i$=INKEY$
447 eS3 WEND
448 ME IF i$="j" THEN ns=1:GOTO NeuesSpiel
449 js IF i$="n" THEN SCREEN CLOSE 2:END
450 sc0 Daten0:
451 293 DATA 4,1,4,1,2,3,2,3
452 18 DATA 3,2,3,2,1,4,1,4
453 4V DATA 2,3,2,3,4,1,4,1
454 3U DATA 1,4,1,4,3,2,3,2
455 Ra0 Daten90:
456 3K3 DATA 3,4,3,4,1,2,1,2
457 AR DATA 2,1,2,1,4,3,4,3
458 9Q DATA 1,2,1,2,3,4,3,4
459 8P DATA 4,3,4,3,2,1,2,1
460 2a0 SUB Anleitung STATIC
461 ML4 FOR x=1 TO 0 STEP -.01

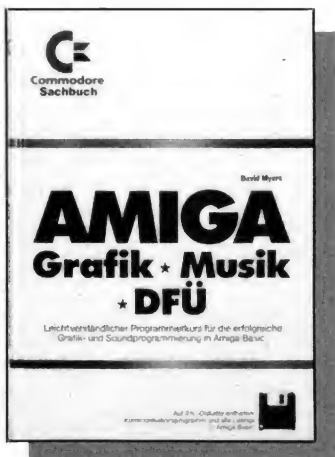
```

```

462 P16 PALETTE 0,x,x,x:PALETTE 1,x,x,x:PALETTE 2,x,x,x
463 Ot4 NEXT x
464 QS CLS:COLOR 2,0
465 1Q PRINT TAB(33)"SPIELANLEITUNG":PRINT
466 Mg PRINT "Shogun ist ein Brettspiel, das zu zweit gespielt
wird."
467 tW PRINT
468 17 PRINT "Jeder Spieler hat acht Steine, von denen der jewe
ils gelb markierte der SHOGUN"
469 fR PRINT "(mit dem König im Schach vergleichbar) ist."
470 b1 PRINT "Die Zahlen auf den Steinen geben an, wieviele Fel
der die Steine ziehen dürfen."
471 Fh PRINT "Wenn ein Stein gesetzt wurde, ändert sich dessen
Zahl nach einem bestimmten"
472 Hy PRINT "System, das hier aber nicht näher erläutert wird,
da das Spiel sonst an Reiz"
473 q7 PRINT "verlieren würde. Auf dem Shogun können die Zahle
n 1 oder 2, auf den anderen"
474 3L PRINT "Steinen die Zahlen 1 bis 4 erscheinen."
475 Ft PRINT "Die Spieler ziehen abwechselnd, Spieler 1 (rot)
beginnt. Ein Zug kann nicht"
476 IZ PRINT "verweigert werden."
477 9v PRINT "Der Spieler klickt den Stein, den er ziehen möcht
e, mit der linken Maustaste"
478 Eo PRINT "an, bewegt den Mauspfeil dann auf das Feld, auf d
as der Stein gesetzt werden"
479 m1 PRINT "soll und drückt erneut die linke Maustaste."
480 tT PRINT "Hat ein Spieler eine Figur angeklickt, die er doc
h nicht setzen möchte, kann"
481 rf PRINT "er sie durch nochmaliges Anklicken wieder loslas
sen. Ein bereits vollständig"
482 oO PRINT "ausgeführter Zug kann jedoch nicht zurückgenommen
werden!"
483 y8 PRINT:PRINT
484 Ze PRINT TAB(33)"Weiter mit W"
485 Eq FOR x=0 TO 1 STEP .01
486 VK6 PALETTE 0,x,x,x:PALETTE 1,x,x,x
487 mH4 NEXT x
488 iA WHILE INKEY$<>"w"
489 K8 WEND
490 po FOR x=1 TO 0 STEP -.01
491 aP6 PALETTE 0,x,x,x:PALETTE 1,x,x,x
492 rM4 NEXT x
493 sI CLS:PRINT:PRINT
494 EW PRINT "Die Steine dürfen nur waagrecht und senkrecht, n
icht diagonal gezogen werden."
495 ib PRINT "Es dürfen weder gegnerische noch eigene Steine üb
ersprungen werden. Ein Stein"
496 v1 PRINT "darf in einem Zug höchstens einmal seine Richtung
um 90 ändern!"
497 nC PRINT "Gegnerische Steine dürfen geschlagen werden."
498 cz PRINT "Der Shogun darf nicht auf ein vom Gegner bedrohte
s Feld gesetzt werden.Wenn der"
499 VX PRINT "Shogun durch einen gegnerischen Stein bedroht wir
d, muß er durch eine der fol-"
500 9G PRINT "genden Möglichkeiten in Sicherheit gebracht wer
den:"
501 kR PRINT "- er wird auf ein nicht bedrohtes Feld gesetzt"
502 OV PRINT "- er schlägt den ihn bedrohenden Stein, wenn dies
er nicht gedeckt ist"
503 80 PRINT "- zwischen ihn und den ihn bedrohenden Stein wird
ein anderer Stein gesetzt"
504 U7 PRINT
505 uZ PRINT "Verloren hat,"
506 j2 PRINT "- wer außer dem Shogun nur noch einen weiteren St
ein besitzt oder"
507 QX PRINT "- wessen Shogun so bedroht wird, daß dieser keine
Möglichkeit mehr hat,sich von"
508 YL PRINT "der Bedrohung zu befreien."
509 ZC PRINT
510 wU PRINT "Und nun viel Spaß beim Spiel!"
511 bE PRINT
512 16 PRINT TAB(33)"Weiter mit W"
513 gI FOR x=0 TO 1 STEP .01
514 xm6 PALETTE 0,x,x,x:PALETTE 1,x,x,x
515 Ej4 NEXT x
516 Ac WHILE INKEY$<>"w"
517 ma WEND
518 OQO END SUB
(C) 1987 M&T

```

Listing 1. (Schluß)



Grafik — Musik — DFÜ

Dieses im Rahmen der Commodore Sachbuchreihe erschienene Buch beschäftigt sich mit der Programmierung von Grafik, Musik und DFÜ auf dem Amiga. Vor dem Einstieg in diese Thematik erfährt der Leser etwas über die Geschichte des Amiga sowie Grundlagen zu den Themen Grafik und Amiga-DOS. Zur Verständlichkeit sowohl dieser als auch der weiteren Abschnitte des Buches tragen Informationskästen bei, in denen zusammengefaßt Grundwissen oder Anmerkungen zu einem bestimmten Thema vermittelt werden.

Der Autor David Myers führt in das gerade auf dem Amiga faszinierende Thema »Grafik« über die von der Programmiersprache Logo bekannte »Turtle Grafik« ein. Auch der »Animation« widmet er ein eigenes Kapitel. Leider bleibt D. Myers an der Oberfläche und geht nur kurz auf die Möglichkeiten ein, die Amiga-Basic mit seinen »Object«-Befehlen bietet.

Der zweite Schwerpunkt des Buches liegt bei der »Musik«. Unter der Überschrift »Sound« erlangt der Leser einiges an Hintergrundwissen zur Theorie der Tonerzeugung auf Computern allgemein, und speziell die Codierung von Tönen im Amiga. Dabei werden auch Begriffe wie »Amplitude« und »ADSR-Hüllkurve« erklärt. Der Leser erhält so einen Überblick über die wichtigsten Fachbegriffe dieses Gebietes. Innerhalb dieses Teiles behandelt David Myers auch das Thema Sprache und die Verwendung von Phonemen.

Ein eigenes Kapitel bildet die Abhandlung über »Synthetische Musik«. Hier stellt der Autor zum Beispiel Tonwellenformen vor. Nebenbei lernt der Leser, wie sich etwa die Titelmu-

sik von »Star Wars« auf dem Amiga programmieren läßt. Wer sich also dem Thema Musik auf dem Amiga widmen will, findet hier eine Vielzahl nützlicher Informationen.

Der letzte Schwerpunkt des Buches befaßt sich mit der Datenfernübertragung (DFÜ). Neben obligatorischem Grundwissen zeigt der Autor dort, wie ein Terminalprogramm in Basic erstellt werden kann. Ergänzend fügt er eine Liste mit während der DFÜ eventuell auftretenden Fehlern und deren Behebung hinzu. Eine Beschreibung wichtiger Mailboxen und Informationsdienste rundet das Kapitel ab.

Wer die Programmierung der drei Gebiete »Grafik — Musik — DFÜ« mit Amiga-Basic lernen will, findet in diesem Buch sicher geeignete Unterstützung, zumal es zu den einzelnen Themen Programmbeispiele enthält. Diese liegen dem Buch auf einer Diskette bei.

(Ingolf Krüger/rs)

Amiga Grafik — Musik — DFÜ; David Myers; Markt & Technik Verlag AG; 225 Seiten; ISBN 3-89090-579-X; Preis 59 Mark

Die fraktale Geometrie der Natur

Wer kennt sie nicht, die »Apfelmännchen«. Unter diesem Namen wurden die selbstähnlichen Abbildungen (eine Untergruppe der »Fraktale«) einem größeren Publikum bekannt. Gerade auf dem Amiga existieren eine Vielzahl von Programmen, die diese teilweise faszinierenden Grafiken erzeugen. Entdeckt wurden Sie von Benoit Mandelbrot, seines Zeichens Professor für Mathematik an der Harvard Universität. Unter dem Titel »Die fraktale Geometrie der Natur« beschreibt er die den Fraktalen zugrunde liegende Theorie und definiert eine Vielzahl von Begriffen. Sein Ziel ist die Verknüpfung natürlicher Strukturen mit seinem Prinzip der Selbstähnlichkeit. Der Hintergrund der Fraktale ist jedoch (oft recht aufwendige) Mathe-



matik. Und so behandelt dieses Buch auch nicht die Umsetzung der verschiedenen Fraktale auf den Computer, sondern schildert deren mathematischen Hintergrund. Es enthält eine Vielzahl von Abbildungen, die teilweise so fantastisch realistisch sind, daß sie dem Leser eine nähere Beschäftigung mit dem Begriff Fraktal nahezu aufzwingen. Die Übersetzung aus dem Englischen ist gut und flüssig zu lesen. Nach einer kurzen Einleitung und der Vorstellung erster Fraktale und deren Problemstellungen führt Benoit B. Mandelbrot den Leser zu Themen wie »Galaxien und Wirbel«, »skaleninvariante Fraktale« und »nichtskaleninvariante Fraktale«. Ausführliche Quellen-, Literatur- und Stichwortverzeichnisse runden das umfangreiche Werk ab.

Dieses Buch ist nicht dazu geeignet, auf die schnelle den Schlüssel zur Umsetzung von Apfelmännchen und anderen fraktalen Strukturen auf den heimischen Computer zu liefern. Wer sich jedoch mit dem nötigen mathematischen Grundwissen und (wissenschaftlichem) Interesse an das Buch heranwagt, wird schon wegen der zahlreichen Abbildungen und der vielfältigen Erscheinungsformen der Fraktale das Buch schätzen.

(Ingolf Krüger/rs)

Die fraktale Geometrie der Natur; Benoit B. Mandelbrot; Birkhäuser Verlag; 491 Seiten; ISBN 3-7643-1771-X; Preis 118 Mark

Das große Amiga-Public Domain-Buch

Dieses Buch ist allen Amiga-Benutzern gewidmet, die sich näher mit dem breiten Spektrum der Public Domain-Programme beschäftigen möchten. Es gibt Hilfestellung bei vielen Fragen, die zu diesem Thema auftreten.

In ersten Teil des Buchs wird durch eine Begriffserläuterung erklärt, was es mit den Bezeichnungen »Public Domain« und »Shareware« auf sich hat und welche Regeln beim Umgang mit Public Domain-Disketten zu beachten sind. Des weiteren folgt eine kurze Unterweisung im Umgang mit dem Amiga. Es werden aber nur die nötigsten Handgriffe beschrieben.

Der zweite Teil, der mit etwa 200 Seiten am umfangreichsten ausgefallen ist, enthält sehr ausführliche deutsche



Beschreibungen, Anleitungen und Quellenangaben zu 43 PD-Programmen. Dabei gliedert sich jede Programmbeschreibung in die Erklärung des Programmaufrufs, den Zweck der jeweiligen Routine, die Herkunftsbestimmung und die eigentliche Bedienung des Programms. Dabei ist die Erklärung sehr ausführlich geworden. Einige Programme wie beispielsweise das Rollenspiel »Hack« oder das DFÜ-Programm »Kermit« werden auf jeweils über zehn Buchseiten beschrieben. Der in Englisch nicht so bewanderte Anwender erhält somit die Chance, komplexere PD-Programme besser auszunutzen, da ihm genauere Informationen zum jeweiligen Programm offeriert werden.

In dritten Teil des ersten Bandes, der mit »Listen« umschrieben ist, findet der Käufer ein Inhaltsverzeichnis von drei PD-Serien: Fish 1 bis 127, Panorama 1 bis 17b und FAUG 1 bis 39. Das Verzeichnis, in dem zu jedem Programm ein Kurzkomentar enthalten ist, findet sich in dreifacher Ausführung: einmal ist es nach Themen sortiert, ein anderes Mal alphabetisch geordnet und in der dritten Ausführung nach Diskettenserien gegliedert. Damit findet jeder Leser schnell das Programm und die Diskettennummer der für ihn interessanten Routine.

Weitere Bücher dieser Serie sind bereits in Arbeit, so daß sich jeder Interessent eine sehr sinnvolle und hilfreiche Bibliothek in sein Regal stellen kann. Der zweite Band ist seit Herbst 1988 auf dem deutschen Markt, eine Besprechung folgt.

(Jens Krumbeck/rs)

Das große Amiga Public Domain Buch — Band 1, Stefan Ram/Jens A. Hertwig, technicSupport-Verlag, 350 Seiten, ISBN 3-926847-01-8, Preis 49 Mark

Amiga 3D-Grafik und Animation

Um anspruchsvolle 3D-Grafiken zu erstellen, ist sowohl mathematisches Grundwissen als auch eine Kenntnis der Grundlagen zur Erzeugung von Grafiken mit dem Computer notwendig. Beides vermittelt auf anschauliche Weise das Buch »Amiga 3D-Grafik und Animation« von Axel Plenge. Nach einer kurzen Einführung stellt der Autor bereits die wichtigsten Grundlagen zum Aufbau einer Farbgrafik auf dem Amiga vor, ohne sich jedoch in technischen Fachwörtern und ausschweifenden Schilderungen zu ergehen. Im dritten Kapitel werden sodann die Grundlagen für die zweidimensionale Grafik gelegt. In diesem Kapitel geht der Autor auch in puncto Mathematik »in medias res«. Doch alle mathematischen Grundlagen werden auf eine sehr anschauliche und dadurch verständliche Weise dargebracht. Im Anhang findet sich zusätzlich ein Abschnitt, der die für die Computergrafik wichtige Mathematik (Winkelfunktionen wie Sinus, Cosinus und Matrizenrechnung) wiederholt. Selbst Mathematikmuffel werden kaum mit Verständnisproblemen zu kämpfen haben.

Neben dem mathematischen Grundwissen erfährt der Leser bereits in Kapitel 3 von Themen wie Vergrößerung/Verkleinerung, Spiegelung, Drehung, Translation und Clipping von grafischen Darstellungen. Kapitel 4 bringt dann den Einstieg in die dreidimensionale Bildschirmgrafik. Der Autor leitet nach einem Ausflug in die »Welt« der Koordinatensysteme und mathematischen Grundlagen von Transformationen zweidimensionaler Grafik (Spiegelung, Drehung...) über auf den dreidimensionalen Raum.

Kapitel 5 widmet sich dem Problem der verdeckten Linien und Flächen. Daran schließt sich ein Kapitel an, das vollständig dem Thema Ray-Tracing gewidmet ist. Den Abschluß bildet die Behandlung von Rotationskörpern. Das durchweg gelungene Werk wird durch den bereits erwähnten Anhang, der auch eine Schilderung der verwendeten Library-Routinen enthält, zahlreiche Literaturhinweise und ein ausführliches Stichwortverzeichnis abgerundet.

(Ingolf Krüger/rs)

Amiga 3D-Grafik und Animation; Axel Plenge; Markt & Technik Verlag AG; 376 Seiten; ISBN 3-89090-526-9; Preis 69 Mark

Computer — die leisen Eroberer



Wie sie funktionieren, wo ihre Grenzen liegen.

Ob im Büro oder Zuhause — der Computer hat unwiderruflich seinen Platz in unserer Gesellschaft eingenommen. Doch Angst und Mißtrauen begleiten den lautlosen Einzug in unser Leben.

Klaus Kupfernagel will dem mit seinem Buch entgegenwirken. »Der Computer«, schreibt er, »wird unsere Gesellschaft verändern, aber das Wissen dieser Gesellschaft über die Computer ist gering.« Kupfernagel erläutert gezielt Funktion, Möglichkeiten, Auswirkungen und Grenzen des Schreckgespenstes »Denkmaschine«. Er vermittelt somit das nötige Wissen für die erfolgreiche Auseinandersetzung mit den Konsequenzen und Gefahren der Computer.

Das Buch ist durchaus kein Lehrbuch. Allein die lockere, witzig-spritzige Art und Weise, in der das Buch geschrieben ist, macht es lesenswert. Viele anschauliche Beispiele bringen auch Einsteigern die Bits und Bytes, RAMs und ROMs,

Floppys, Zentraleinheiten und andere Feinheiten nahe.

(Jan-Dirk Mittmann/rs)

Computer — die leisen Eroberer, Klaus Kupfernagel, Markt & Technik Verlag, 270 Seiten, ISBN 3-89090-179-4, 14,80 Mark

Grafik auf dem Amiga

»Grafik auf dem Amiga« ist in fünf Teile gegliedert, die unabhängig voneinander gelesen werden können. Der Einsteiger kann sich zunächst den Grundlagen der Grafik zuwenden. Er erfährt hier etwas über grundsätzliche Techniken der Bilderzeugung auf einem Monitor/Fernseher und gelangt unversehens in die mathematisch/geometrische Ableitung von Algorithmen zur Erzeugung von Linien, Kreisen und dreidimensionalen Objekten.

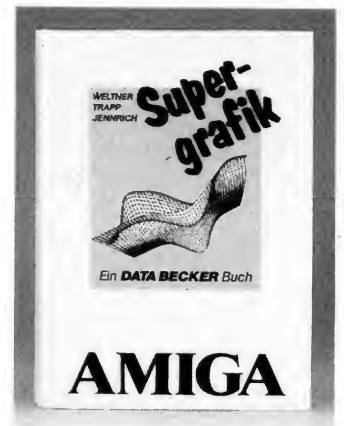
Die Hardwarespezialisten finden im folgenden Kapitel eine Beschreibung der Amiga-Hardware.

Der dritte Teil beschäftigt sich mit den Möglichkeiten des Amiga-Basic, die Grafikhardware zu beeinflussen. Da Grundkenntnisse der Programmiersprache vorausgesetzt werden, hält sich der Autor auch nicht lange mit den grundsätzlichen Techniken auf, sondern gelangt recht schnell in das interessante Gebiet der Computeranimation, welches dann um so ausführlicher behandelt wird.

Ein weiterer Teil führt in die grundsätzliche Arbeitsweise und Struktur des Betriebssystems beziehungsweise der von der Grafikdarstellung betroffenen Systembibliotheken ein. Wer mehr über die Funktionsweise der Amiga-Grafik wissen möchte, kommt um dieses Buch nicht herum.

(Joschy Polierer/rs)

Grafik auf dem Amiga, Manfred Köhlen, Verlag Markt & Technik AG, 340 Seiten, ISBN 3-89090-236-7, Preis: 49 Mark



Supergrafik

Unbestritten lassen sich mit dem Amiga fantastische Grafiken auf den Bildschirm zaubern. Wer solche Kunstwerke allerdings selbst programmieren möchte, der benötigt fundierte Kenntnisse über die Systemgrafikroutinen des Amiga. Dieses Wissen vermittelt das vorliegende Buch.

Je nach Vorkenntnissen bietet die Aufteilung des Werkes jeweils drei Einstiegspunkte für Anfänger, Fortgeschrittene und angehende Profis. Im ersten Teil wird der Anfänger mit den elementaren Grafikbefehlen des Amiga-Basic bekannt gemacht. Er lernt, wie sich Grafiken mit den Mitteln, die Basic zur Verfügung stellt, effektiv erstellen lassen. Im zweiten Teil wird dem fortgeschrittenen Basic-Programmierer gezeigt, wie er die Grafikroutinen des Betriebssystems in Basic-Programmen nutzen kann. Mit deren Hilfe ist es dann möglich, von Basic aus den 64farbigen Halfbrite- und den 4096farbigen H.A.M.-Modus einzuschalten, den Coprozessor Copper zu programmieren und Grafik auf dem Drucker im Multitasking-Betrieb auszugeben. Der dritte Teil wendet sich an die C-Programmierer und zeigt, wie man Screens und Fenster in C erstellt, wie Sprites, BOBs und VSprites programmiert und dann mit Hilfe des vollautomatischen Animationssystems des Amiga in Bewegung versetzt werden.

»Supergrafik« enthält alle wichtigen Informationen, die ein Programmierer benötigt, wenn er in die Welt der Amiga-Grafik einsteigen will. Es kann deshalb jedem, der sich für die Grafikprogrammierung mit dem Amiga interessiert, empfohlen werden.

(Christian Schneider/rs)

Supergrafik, Weltner/Trapp/Jennrich, Data Becker, 690 Seiten, ISBN 3-89011-254-4, Preis: 59 Mark



Sprite-Editor

Das Programm erkennt, ob es sich um ein »AOE«-Objekt (Amiga-Object-Editor-Objekt), oder um einen DPaint-Brush handelt. DPaint-Brushes müssen in der Größe von 32 x 32 Punkten mit 16 Farben vorliegen. Wollen Sie selbsterstellte Sprites auf Diskette ablegen, so wählen Sie »SPEICHERN«. Hier sind die Nummern des ersten und des letzten zu speichernden Objekts angegeben. Es können also beliebig viele aufeinanderfolgende Sprites in einem File gespeichert werden. Wollen Sie nur ein Sprite speichern, geben Sie einfach zwei gleiche Werte an. Nach Eingabe des Dateinamens werden die Objekte im »AOE«-Format gespeichert. Den Aufbau eines »AEO«-Files sehen Sie in Tabelle 1.

Zahlreiche Icons erleichtern die Arbeit

Icon 2 (CLR): Wenn Sie dieses Icon anwählen, wird das aktuelle Sprite gelöscht.

Icon 3 (zwei waagerechte Pfeile): Horizontales Spiegeln des aktuellen Sprites.

Icon 4 (zwei senkrechte Pfeile): Vertikales Spiegeln des aktuellen Sprites.

Icon 5 (geknickter Pfeil): Drehen des aktuellen Sprites um 90 Grad.

Icon 6 (zwei sich überlappende Quadrate): Kopieren von Sprites. Neben der Quell- und Zielnummer des Objekts kann beim Kopieren von mehreren aufeinanderfolgenden Sprites auch die gewünschte Anzahl eingegeben werden.

Kopieren und Animieren von Sprites

Icon 7 (»i«): Info zum Programm.

Icon 8 (Bildprojektor-Symbol): Animation von mehreren Sprites. Geben Sie die Nummern des ersten und letzten Sprites der Animationsfolge ein und klicken Sie »FERTIG« an. Daraufhin beginnt die Animation, deren Geschwindigkeit mit der Maus verändert werden kann. Die Zahl zwischen den Pfeilen gibt die Geschwindigkeit in 1/50-Sekunden an.

Haben Sie Ihre Arbeit beendet und wollen den Editor verlassen, bewegen Sie die Maus in die rechte obere Bildschirmcke und drücken den rechten Mausknopf.

Nun steht dem munteren Erstellen von Sprites und Anima-

tionssequenzen nichts mehr im Wege. Grenzen setzt nur die eigene Kreativität — und von der besitzen Amiga-Besitzer ja eine ganze Menge.

(Michael Bertsch/C.
Ewald/M.Jobst/rs)

So geben Sie das Listing ein

Der Objekt-Editor ist in Maschinensprache (Assembler) geschrieben. Um jedem Amiga-Besitzer die Eingabe des Listings zu ermöglichen, haben wir das Programm für Sie umgewandelt. Die abgedruckte Version ist mit Amiga-Basic einzugeben. Diesen Interpreter finden Sie auf der Extras-Diskette, die bei der Auslieferung jedem Amiga beiliegt. Das Programm erzeugt beim Start das lauffähige Programm mit Namen »AOE_V1.2« auf der eingelegten Diskette. Bitte geben Sie das Programm unbedingt mit Checkie 42 (Seite 159) ein. Die Hinweise zum Checksummer sollten Sie unbedingt beachten.

Programmname: AOE_Gen

Computer: Amiga 500,1000,2000 mit Kickstart 1.2

Sprache: Amiga-Basic 1.2

```
1 tIO REM Generiert Programm AOE_V1.2
2 ag CLS
3 gO OPEN "AOE_V1.2" FOR OUTPUT AS 1
4 BS READ anz
5 oa FOR i=1 TO anz
6 3n1 READ h$
7 yB2 wert1=ASC(LEFT$(h$,1))
8 FD IF wert1>64 THEN wert1=wert1-55 ELSE wert1=wert1-48
9 FI wert1=wert1*16
10 7c wert2=ASC(RIGHT$(h$,1))
11 ad IF wert2>64 THEN wert2=wert2-55 ELSE wert2=wert2-48
12 P1 wert=wert1+wert2
13 9G PRINT #1,CHR$(wert);
```

```
14 JOO NEXT
15 3n CLOSE 1
16 Ov END
17 R4 DATA 19072
18 ie DATA 00,00,03,F3,00,00,00,00,00,00
19 aj DATA 00,02,00,00,00,00,00,00,00,01
20 UF DATA 00,00,0F,1A,00,00,01,00,00,00
21 gb DATA 03,E9,00,00,0F,1A,4E,F9,00,00
22 72 DATA 2B,94,2C,78,00,04,43,F9,00,00
23 ut DATA 3B,30,42,80,4E,AE,FD,D8,23,C0
24 Sy DATA 00,00,3A,3E,43,F9,00,00,3B,41
25 az DATA 42,80,4E,AE,FD,D8,23,C0,00,00
26 6d DATA 3A,42,43,F9,00,00,3B,4D,42,80
27 6p DATA 4E,AE,FE,D8,23,C0,00,00,3A,46
28 N4 DATA 4E,75,2C,78,00,04,22,79,00,00
29 D7 DATA 3A,3E,4E,AE,FE,62,22,79,00,00
30 Te DATA 3A,42,4E,AE,FE,62,22,79,00,00
31 bf DATA 3A,46,4E,AE,FE,62,4E,75,2C,79
32 KH DATA 00,00,3A,46,4E,AE,FF,B2,41,F9
33 gd DATA 00,00,2D,46,4E,AE,FF,3A,23,C0
34 9F DATA 00,00,2D,84,23,C0,00,00,2D,B4
35 et DATA 23,C0,00,00,2D,E4,41,F9,00,00
36 PL DATA 2D,66,4E,AE,FF,3A,23,C0,00,00
37 fb DATA 3A,52,20,40,4E,AE,FE,D4,23,C0
38 eK DATA 00,00,3A,62,20,79,00,00,3A,52
39 W3 DATA 23,E8,00,32,00,00,3A,66,23,E8
40 BS DATA 00,32,00,00,3A,6E,20,68,00,32
41 ge DATA 23,E8,00,04,00,00,3A,5A,2C,79
42 4I DATA 00,00,3A,3E,20,79,00,00,3A,62
43 v1 DATA 43,F9,00,00,32,6E,30,3C,00,20
44 2u DATA 4E,AE,FF,40,2C,79,00,00,3A,46
45 oE DATA 4E,AE,FE,80,4E,75,2C,79,00,00
46 kU DATA 3A,46,20,79,00,00,3A,52,4E,AE
47 QW DATA FF,B8,20,79,00,00,2D,84,4E,AE
48 63 DATA FF,BE,4E,AE,FF,2E,4E,75,2C,78
49 2p DATA 00,04,93,C9,4E,AE,FE,DA,28,40
```

```
50 yZ DATA 4A,AC,00,AC,66,00,00,18,41,EC
51 pR DATA 00,5C,4E,AE,FE,80,41,EC,00,5C
52 hO DATA 4E,AE,FE,8C,23,C0,00,00,3A,82
53 aC DATA 4E,75,4A,B9,00,00,3A,82,67,00
54 A1 DATA 00,0C,22,79,00,00,3A,82,4E,AE
55 HD DATA FE,86,42,80,4E,75,2C,79,00,00
56 bd DATA 3A,3E,43,F9,00,00,3A,98,4E,AE
57 71 DATA FF,3A,23,FC,00,00,3B,08,00,00
58 c1 DATA 3A,9C,22,79,00,00,3A,66,30,3C
59 fZ DATA 00,10,4E,AE,FE,AA,22,79,00,00
60 8J DATA 3A,66,42,80,42,81,34,3C,00,E0
61 WD DATA 36,3C,00,E0,4E,AE,FE,CE,22,79
62 Dd DATA 00,00,3A,66,30,3C,00,02,4E,AE
63 uQ DATA FE,AA,22,79,00,00,3A,66,42,80
64 RH DATA 4E,AE,FE,A4,22,79,00,00,3A,66
65 JK DATA 10,3C,00,00,4E,AE,FE,9E,42,86
66 H1 DATA 22,79,00,00,3A,66,33,46,00,24
67 nV DATA 33,7C,00,00,00,26,30,06,32,3C
68 xL DATA 00,E0,4E,AE,FF,0A,22,79,00,00
69 DZ DATA 3A,66,33,7C,00,00,00,24,33,46
70 HO DATA 00,26,30,3C,00,E0,32,06,4E,AE
71 AZ DATA FF,0A,DC,7C,00,07,BC,7C,00,E7
72 9q DATA 66,C2,22,79,00,00,3A,66,30,3C
73 CF DATA 00,04,4E,AE,FE,AA,22,79,00,00
74 1U DATA 3A,66,33,7C,00,70,00,24,33,7C
75 KC DATA 00,01,00,26,30,3C,00,70,32,3C
76 85 DATA 00,DF,4E,AE,FF,0A,22,79,00,00
77 dG DATA 3A,66,33,7C,00,01,00,24,33,7C
78 tV DATA 00,70,00,26,30,3C,00,DF,32,3C
79 OA DATA 00,70,4E,AE,FF,0A,22,79,00,00
80 BR DATA 3A,66,30,3C,00,01,4E,AE,FE,AA
81 08 DATA 22,79,00,00,3A,66,30,3C,00,E1
82 E1 DATA 42,81,34,3C,01,3F,36,3C,00,FF
83 UT DATA 4E,AE,FE,CE,22,79,00,00,3A,66
84 8H DATA 30,3C,00,00,32,3C,00,E1,34,3C
85 d1 DATA 00,E1,36,3C,00,FF,4E,AE,FE,CE
86 oc DATA 22,79,00,00,3A,66,30,3C,00,02
87 EH DATA 4E,AE,FE,AA,22,79,00,00,3A,66
```

```
88 31 DATA 30,3C,00,01,4E,AE,FE,A4,30,3C
89 nL DATA 00,02,38,3C,00,FF,3A,3C,00,07
90 H1 DATA 3C,3C,01,20,3E,3C,00,28,4E,B9
91 VC DATA 00,00,04,88,30,3C,00,04,38,3C
92 QS DATA 00,FE,3A,3C,00,06,3C,3C,01,21
93 R5 DATA 3E,3C,00,29,4E,B9,00,00,04,88
94 qd DATA 22,79,00,00,3A,66,30,3C,00,10
95 MP DATA 4E,AE,FE,AA,22,79,00,00,3A,66
96 p1 DATA 30,3C,01,00,32,3C,00,08,34,3C
97 v5 DATA 01,1F,36,3C,00,27,4E,AE,FE,CE
98 fw DATA 4B,F9,00,00,32,AE,3C,3C,00,10
99 tP DATA 30,06,22,79,00,00,3A,66,4E,AE
100 jx DATA FE,AA,30,1D,32,1D,34,1D,36,1D
101 Xn DATA 22,79,00,00,3A,66,4E,AE,FE,CE
102 II DATA DC,7C,00,01,BC,7C,00,20,66,D8
103 KE DATA 30,3C,00,02,38,3C,00,0F,3A,3C
104 1U DATA 00,E7,3C,3C,00,01,3E,3C,00,F9
105 4E DATA 4E,B9,00,00,04,88,30,3C,00,04
106 Zk DATA 38,3C,00,0E,3A,3C,00,E6,3C,3C
107 O1 DATA 00,1C,3E,3C,00,FA,4E,B9,00,00
108 Uh DATA 04,88,2C,79,00,00,3A,46,20,79
109 WR DATA 00,00,3A,66,43,F9,00,00,34,36
110 mH DATA 30,3C,01,09,32,3C,00,50,4E,AE
111 TF DATA FF,8E,43,F9,00,00,34,FE,20,79
112 D2 DATA 00,00,3A,66,30,3C,01,00,32,3C
113 Ue DATA 00,66,4E,AE,FF,8E,43,F9,00,00
114 an DATA 35,9E,20,79,00,00,3A,66,30,3C
115 FH DATA 00,F8,32,3C,00,BC,4E,AE,FE,8E
116 NL DATA 43,F9,00,00,35,C6,20,79,00,00
117 pZ DATA 3A,66,30,3C,00,FB,32,3C,00,E2
118 D1 DATA 4E,AE,FF,8E,4B,F9,00,00,3B,5F
119 Rn DATA 3A,3C,00,0A,3C,3C,00,E8,3E,3C
120 QW DATA 00,B4,4E,B9,00,00,04,18,4B,F9
121 Ts DATA 00,00,3B,74,3A,3C,00,03,3C,3C
122 EI DATA 01,03,3E,3C,00,C2,4E,B9,00,00
123 Eh DATA 04,18,4B,F9,00,00,3B,69,3A,3C
124 yJ DATA 00,05,3C,3C,00,FB,3E,3C,00,DE
125 21 DATA 4E,B9,00,00,04,18,4E,B9,00,00
```



```

126 R3 DATA 05,C2,4E,75,2C,79,00,00,3A,3E
127 z1 DATA 22,79,00,00,3A,6E,33,46,00,24
128 ps DATA 33,47,00,26,30,3C,00,01,4E,AE
129 G3 DATA FE,9E,22,79,00,00,3A,6E,42,80
130 8Q DATA 4E,AE,FE,AA,22,79,00,00,3A,6E
131 mS DATA 20,4D,30,05,4E,AE,FF,C4,22,79
132 GK DATA 00,00,3A,6E,DC,7C,00,01,DE,7C
133 Db DATA 00,01,33,46,00,24,33,47,00,26
134 b4 DATA 42,80,4E,AE,FE,9E,22,79,00,00
135 cw DATA 3A,6E,30,3C,00,03,4E,AE,FE,AA
136 up DATA 22,79,00,00,3A,6E,20,4D,30,05
137 YL DATA 4E,AE,FF,C4,4E,75,2C,79,00,00
138 07 DATA 3A,3E,22,79,00,00,3A,6E,4E,AE
139 gw DATA FE,AA,22,79,00,00,3A,6E,33,44
140 Ra DATA 00,24,33,45,00,26,30,06,32,05
141 j8 DATA 4E,AE,FF,0A,22,79,00,00,3A,6E
142 Qb DATA 30,06,32,07,4E,AE,FF,0A,22,79
143 wJ DATA 00,00,3A,6E,30,04,32,07,4E,AE
144 5y DATA FF,0A,22,79,00,00,3A,6E,30,04
145 S1 DATA 32,05,4E,AE,FF,0A,4E,75,42,87
146 yx DATA 41,F9,00,00,32,FE,00,18,32,18
147 3p DATA 34,18,36,18,B0,45,62,00,00,1A
148 j3 DATA B2,46,62,00,00,14,B4,45,65,00
149 no DATA 00,0E,B6,46,65,00,00,08,4E,F9
150 XR DATA 00,00,05,20,DE,7C,00,01,BE,7C
151 ds DATA 00,10,66,D0,08,39,00,06,00,BF
152 HY DATA E0,01,67,F6,60,00,26,FE,23,F9
153 C9 DATA 00,00,3A,66,00,00,3A,6E,33,C7
154 PP DATA 00,00,3A,74,3E,39,00,00,3A,72
155 1v DATA CE,FC,00,08,28,47,D9,FC,00,00
156 N8 DATA 32,AE,61,00,00,62,61,00,00,68
157 00 DATA 30,3C,00,01,61,00,FF,38,61,00
158 y2 DATA 00,5C,30,3C,00,01,61,00,FF,2C
159 rd DATA 42,87,3E,39,00,00,3A,74,33,C7
160 ug DATA 00,00,3A,72,CE,FC,00,08,28,47
161 SN DATA D9,FC,00,00,32,AE,61,00,00,2C
162 AR DATA 61,00,00,32,30,3C,00,02,61,00
163 Fn DATA FF,02,61,00,00,26,30,3C,00,04
164 b7 DATA 61,00,FE,F6,61,00,00,2C,08,39
165 Bq DATA 00,06,00,BF,E0,01,67,F6,60,00
166 Bq DATA 26,78,38,1C,3A,1C,3C,1C,3E,1C
167 CH DATA 4E,75,98,7C,00,01,9A,7C,00,01
168 pz DATA DC,7C,00,01,DE,7C,00,01,4E,75
169 TU DATA 42,80,30,39,00,00,3A,72,C0,FC
170 HY DATA 00,02,22,40,D3,FC,00,00,32,8E
171 8N DATA 30,11,3E,00,61,00,00,48,13,C0
172 98 DATA 00,00,3B,72,30,07,E8,48,61,00
173 wu DATA 00,3A,13,C0,00,00,3B,70,30,07
174 DW DATA E0,48,61,00,00,2C,13,C0,00,00
175 HS DATA 3B,6E,4B,F9,00,00,3B,6E,3A,3C
176 Vv DATA 00,06,3C,3C,00,FB,3E,3C,00,F2
177 EP DATA 23,F9,00,00,3A,66,00,00,3A,6E
178 vb DATA 4E,B9,00,00,04,18,4E,75,C0,7C
179 x3 DATA 00,0F,D0,7C,00,30,B0,7C,00,3A
180 IO DATA 65,00,00,06,D0,7C,00,07,4E,75
181 Ao DATA 2C,79,00,00,3A,7C,00,08,4E,07
182 sq DATA 8C,FC,00,07,CA,BC,00,00,FF,FF
183 yR DATA CC,BC,00,00,FF,FF,20,05,22,06
184 or DATA C0,FC,00,07,C2,FC,00,07,D0,7C
185 1q DATA 00,01,D2,7C,00,01,34,00,36,01
186 D4 DATA D4,7C,00,05,D6,7C,00,05,48,E7
187 jZ DATA F0,00,30,39,00,00,3A,72,D0,7C
188 jB DATA 00,10,22,79,00,00,3A,66,4E,AE
189 cN DATA FE,AA,22,79,00,00,3A,66,4C,DF
190 4G DATA 00,0F,4E,AE,FE,CE,43,F9,00,00
191 Vw DATA 3A,98,30,39,00,00,3A,72,D0,7C
192 nK DATA 00,10,4E,AE,FE,AA,43,F9,00,00
193 Vs DATA 3A,98,30,05,32,06,4E,AE,FE,BC
194 Bc DATA 22,79,00,00,3A,66,30,05,32,06
195 EJ DATA D0,7C,01,00,D2,7C,00,08,4E,AE
196 eX DATA FE,BC,4E,75,42,87,41,F9,00,00
197 Hd DATA 33,2E,30,18,32,18,3A,18,36,18
198 jP DATA B0,45,62,00,00,1A,B2,46,62,00
199 Bz DATA 00,14,B4,45,65,00,00,0E,B6,46
200 gY DATA 65,00,00,08,4E,F9,00,00,07,1A
201 qR DATA DE,7C,00,01,BE,7C,00,16,66,D0
202 4C DATA 08,39,00,06,00,BF,E0,01,67,F6
203 qn DATA 60,00,25,04,CE,FC,00,04,20,47
204 jC DATA D1,FC,00,00,33,DE,22,50,4E,D1
205 7F DATA 08,39,00,06,00,BF,E0,01,67,F6
206 9I DATA 23,F9,00,00,3A,66,00,00,3A,6E
207 hk DATA 60,00,24,DC,0C,79,00,01,00,00
208 2D DATA 3A,7C,67,00,01,84,20,79,00,00
209 DM DATA 3B,10,61,00,00,26,20,79,00,00

210 DI DATA 3B,14,61,00,00,1C,20,79,00,00
211 nL DATA 3B,18,61,00,00,12,20,79,00,00
212 eY DATA 3B,1C,61,00,00,08,61,00,20,2E
213 Vd DATA 60,AE,2E,28,00,7C,24,48,22,48
214 7v DATA D3,FC,00,00,00,80,D1,FC,00,00
215 hI DATA 00,7C,20,3C,00,00,00,1E,23,20
216 5d DATA 51,C8,FF,FC,24,87,4E,75,0C,79
217 bR DATA 00,01,00,00,3A,7C,67,00,01,78
218 3Q DATA 20,79,00,00,3B,10,61,00,00,28
219 Fs DATA 20,79,00,00,3B,14,61,00,00,1E
220 5V DATA 20,79,00,00,3B,18,61,00,00,14
221 eP DATA 20,79,00,00,3B,1C,61,00,00,0A
222 hU DATA 61,00,1F,D0,60,00,FF,50,2E,10
223 93 DATA 24,48,22,48,D1,FC,00,00,00,04
224 2z DATA 20,3C,00,00,00,1E,22,D8,51,C8
225 0x DATA FF,FC,25,47,00,7C,4E,75,0C,79
226 j1 DATA 00,01,00,00,3A,7C,67,00,01,6E
227 CZ DATA 20,79,00,00,3B,10,61,00,00,28
228 01 DATA 20,79,00,00,3B,14,61,00,00,1E
229 Ee DATA 20,79,00,00,3B,18,61,00,00,14
230 1Y DATA 20,79,00,00,3B,1C,61,00,00,0A
231 s2 DATA 61,00,1F,76,60,00,FE,F6,20,3C
232 Hb DATA 00,00,00,1F,22,10,E2,99,20,C1
233 Pc DATA 51,C8,FF,FC,4E,75,0C,79,00,01
234 9z DATA 00,00,3A,7C,67,00,01,62,20,79
235 ea DATA 00,00,3B,10,61,00,00,28,20,79
236 am DATA 00,00,3B,14,61,00,00,1E,20,79
237 dc DATA 00,00,3B,18,61,00,00,14,20,79
238 2t DATA 00,00,3B,1C,61,00,00,0A,61,00
239 1U DATA 1F,28,60,00,FE,A8,20,3C,00,00
240 GH DATA 00,1F,22,10,E3,99,20,C1,51,C8
241 rH DATA FF,F8,4E,75,0C,79,00,01,00,00
242 x4 DATA 3A,7C,67,00,01,5A,2C,79,00,00
243 0Q DATA 3A,46,20,79,00,00,3A,66,43,F9
244 fH DATA 00,00,34,9A,20,3C,00,00,01,09
245 wt DATA 22,3C,00,00,00,50,4E,AE,FF,8E
246 tN DATA 33,FC,00,01,00,00,3A,7C,60,00
247 fJ DATA FE,5C,20,79,00,00,3B,10,61,00
248 u0 DATA 00,28,20,79,00,00,3B,14,61,00
249 Mf DATA 00,1E,20,79,00,00,3B,18,61,00
250 R1 DATA 00,14,20,79,00,00,3B,1C,61,00
251 En DATA 00,0A,61,00,1E,AC,60,00,FE,2C
252 1e DATA 24,48,22,48,D3,FC,00,00,00,40
253 FU DATA D1,FC,00,00,00,44,20,3C,00,00
254 AM DATA 00,0F,22,D8,51,C8,FF,FC,42,AA
255 dD DATA 00,7C,4E,75,20,79,00,00,3B,10
256 s1 DATA 61,00,00,28,20,79,00,00,3B,14
257 X1 DATA 61,00,00,1E,20,79,00,00,3B,18
258 ty DATA 61,00,00,14,20,79,00,00,3B,1C
259 Tr DATA 61,00,00,0A,61,00,1E,5A,60,00
260 8J DATA FD,DA,24,48,22,48,D3,FC,00,00
261 Bm DATA 00,3C,D1,FC,00,00,00,40,20,3C
262 zk DATA 00,00,00,0F,21,21,51,C8,FF,FC
263 VJ DATA 42,92,4E,75,20,79,00,00,3B,10
264 0Q DATA 61,00,00,28,20,79,00,00,3B,14
265 fq DATA 61,00,00,1E,20,79,00,00,3B,18
266 16 DATA 61,00,00,14,20,79,00,00,3B,1C
267 Mf DATA 61,00,00,0A,61,00,1E,0A,60,00
268 5U DATA FD,8A,20,3C,00,00,00,1F,22,10
269 8J DATA E3,49,20,C1,51,C8,FF,F8,4E,75
270 tG DATA 20,79,00,00,3B,10,61,00,00,28
271 51 DATA 20,79,00,00,3B,14,61,00,00,1E
272 vL DATA 20,79,00,00,3B,18,61,00,00,14
273 SF DATA 20,79,00,00,3B,1C,61,00,00,0A
274 p1 DATA 61,00,1D,C8,60,00,FD,48,20,3C
275 09 DATA 00,00,00,1F,22,10,48,41,E2,49
276 uL DATA 48,41,20,C1,51,C8,FF,F4,4E,75
277 xr DATA 2C,79,00,00,3A,46,20,79,00,00
278 Me DATA 3A,66,43,F9,00,00,3A,36,20,3C
279 UP DATA 00,00,01,09,22,3C,00,00,00,50
280 4w DATA 4E,AE,FF,8E,33,FC,00,00,00,00
281 RF DATA 3A,7C,60,00,FD,04,41,F9,00,00
282 V1 DATA 2D,96,2C,79,00,00,3A,46,4E,AE
283 Qt DATA FF,34,23,C0,00,00,3A,56,67,00
284 kJ DATA FC,EA,20,40,23,E8,00,32,00,00
285 bU DATA 3A,6A,22,68,00,32,30,3C,00,01
286 W9 DATA 2C,79,00,00,3A,3E,4E,AE,FE,AA
287 VT DATA 22,79,00,00,3A,6A,30,3C,00,01
288 wx DATA 4E,AE,FE,A4,22,79,00,00,3A,6A
289 M9 DATA 30,3C,00,00,32,3C,00,00,34,3C
290 po DATA 00,DF,36,3C,00,DF,4E,AE,FE,CE
291 3N DATA 23,F9,00,00,3A,6A,00,00,3A,6E
292 EZ DATA 4B,F9,00,00,3B,77,3A,3C,00,05
293 BE DATA 3C,3C,00,5C,3E,3C,00,5B,4E,B9

294 4p DATA 00,00,04,18,4B,F9,00,00,3B,7C
295 u2 DATA 3A,3C,00,09,3C,3C,00,4C,3E,3C
296 q1 DATA 00,6D,4E,B9,00,00,04,18,4B,F9
297 1s DATA 00,00,3C,13,3A,3C,00,07,3C,3C
298 6I DATA 00,54,3E,3C,00,7F,4E,B9,00,00
299 9D DATA 04,18,30,3C,00,02,38,3C,00,46
300 IO DATA 3A,3C,00,50,3C,3C,00,9A,3E,3C
301 E8 DATA 00,60,4E,B9,00,00,04,88,30,3C
302 rs DATA 00,02,38,3C,00,46,3A,3C,00,62
303 JG DATA 3C,3C,00,9A,3E,3C,00,72,4E,B9
304 oT DATA 00,00,04,88,30,3C,00,02,38,3C
305 nw DATA 00,46,3A,3C,00,74,3C,3C,00,9A
306 IT DATA 3E,3C,00,84,4E,B9,00,00,04,88
307 1t DATA 08,39,00,06,00,BF,E0,01,67,F6
308 kr DATA 08,39,00,06,00,BF,E0,01,66,F6
309 rg DATA 22,79,00,00,3A,56,3A,29,00,0E
310 fs DATA 3C,29,00,0C,BA,7C,00,46,65,00
311 CX DATA 00,1E,BC,7C,00,50,65,00,00,16
312 Vw DATA BA,7C,00,9A,62,00,00,0E,BC,7C
313 D0 DATA 00,60,62,00,00,06,60,00,00,52
314 zd DATA BA,7C,00,46,65,00,00,1E,BC,7C
315 Jr DATA 00,62,65,00,00,16,BA,7C,00,9A
316 1Y DATA 62,00,00,0E,BC,7C,00,72,62,00
317 eq DATA 00,06,60,00,04,F4,BA,7C,00,46
318 gJ DATA 65,9A,BC,7C,00,74,65,9A,BA,7C
319 g3 DATA 00,9A,62,8E,BC,7C,00,84,62,88
320 eY DATA 2C,79,00,00,3A,46,20,79,00,00
321 1t DATA 3A,56,4E,AE,FF,B8,60,00,FB,70
322 ga DATA 2C,79,00,00,3A,46,20,79,00,00
323 EZ DATA 3A,56,4E,AE,FF,B8,20,79,00,00
324 04 DATA 3A,62,30,3C,00,1F,32,3C,00,0D
325 Fm DATA 3A,01,36,01,2C,79,00,00,3A,3E
326 Rr DATA 4E,AE,FE,E0,20,79,00,00,3A,62
327 I4 DATA 30,3C,00,1E,32,3C,00,03,34,01
328 MR DATA 36,01,4E,AE,FE,E0,2C,79,00,00
329 18 DATA 3A,46,23,FC,00,00,2D,F6,00,00
330 xo DATA 2D,A8,41,F9,00,00,2D,96,4E,AE
331 qW DATA FF,34,23,C0,00,00,3A,56,20,46
332 xf DATA 23,E8,00,32,00,00,3A,6A,23,E8
333 6F DATA 00,32,00,00,3A,6E,22,68,00,32
334 er DATA 30,3C,00,01,2C,79,00,00,3A,3E
335 7L DATA 4E,AE,FE,AA,22,79,00,00,3A,6A
336 T8 DATA 30,3C,00,01,4E,AE,FE,A4,22,79
337 6A DATA 00,00,3A,6A,30,3C,00,00,32,3C
338 Jd DATA 00,00,3A,3C,00,DF,36,3C,00,54
339 Vf DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
340 kh DATA 30,3C,00,00,32,3C,00,55,34,3C
341 m5 DATA 00,5D,36,3C,00,61,4E,AE,FE,CE
342 u0 DATA 22,79,00,00,3A,6A,30,3C,00,81
343 au DATA 32,3C,00,55,34,3C,00,DF,36,3C
344 1S DATA 00,61,4E,AE,FE,CE,22,79,00,00
345 w4 DATA 3A,6A,30,3C,00,00,32,3C,00,61
346 eD DATA 34,3C,00,DF,36,3C,00,72,4E,AE
347 NW DATA FE,CE,22,79,00,00,3A,6A,30,3C
348 eT DATA 00,00,32,3C,00,73,34,3C,00,24
349 WT DATA 36,3C,00,7F,4E,AE,FE,CE,22,79
350 hB DATA 00,00,3A,6A,30,3C,00,B8,32,3C
351 xW DATA 00,73,34,3C,00,DF,36,3C,00,7F
352 1s DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
353 9P DATA 30,3C,00,00,32,3C,00,7F,34,3C
354 rq DATA 00,DF,36,3C,00,DF,4E,AE,FE,CE
355 vC DATA 4B,F9,00,00,3B,85,3A,3C,00,11
356 jP DATA 3C,3C,00,27,3C,00,00,4F,4E,B9
357 Ak DATA 00,00,04,18,4B,F9,00,00,3B,96
358 3x DATA 3A,3C,00,09,3C,3C,00,27,3E,3C
359 ff DATA 00,6B,4E,B9,00,00,04,18,4B,F9
360 Iu DATA 00,00,3B,9F,3A,3C,00,06,3C,3C
361 YB DATA 00,58,3E,3C,00,91,4E,B9,00,00
362 47 DATA 04,18,30,3C,00,02,38,3C,00,54
363 uY DATA 3A,3C,00,87,3C,3C,00,8C,3E,3C
364 Gf DATA 00,96,4E,B9,00,00,04,88,30,39
365 39 DATA 00,00,2E,82,08,00,00,07,66,00
366 Wv DATA 00,36,08,39,00,06,00,BF,E0,01
367 Hh DATA 66,E8,22,79,00,00,3A,56,3A,29
368 30 DATA 00,0E,3C,29,00,0C,BA,7C,00,54
369 t2 DATA 65,D4,BC,7C,00,87,65,CE,BA,7C
370 f6 DATA 00,8C,62,C8,BC,7C,00,96,62,C2
371 8R DATA 60,00,00,0E,30,39,00,00,2E,82
372 g0 DATA 08,00,00,07,66,F4,0C,B9,00,00
373 jV DATA 00,00,00,00,2E,62,66,00,00,0C

```

Listing 1. Der Quellcode des Object-Editors als DATA-Lader. Bitte mit dem Checksummer auf Seite 159 eingeben.

374 K9	DATA 23,FC,00,00,00,01,00,00,2E,62	458 Id	DATA 3A,6A,30,3C,00,01,4E,AE,FE,A4	542 fr	DATA FF,DC,60,00,F2,D2,20,79,00,00
375 XR	DATA 2C,79,00,00,3A,46,20,79,00,00	459 C9	DATA 22,79,00,00,3A,6A,30,3C,00,00	543 FJ	DATA 2D,B4,2C,79,00,00,3A,46,4E,AE
376 kI	DATA 3A,56,4E,AE,FF,B8,23,FC,00,00	460 p0	DATA 32,3C,00,00,34,3C,00,DF,36,3C	544 JJ	DATA FF,A0,60,00,F2,BE,2C,79,00,00
377 gl	DATA 00,00,00,00,2D,A8,2C,79,00,00	461 7L	DATA 00,54,4E,AE,FE,CE,22,79,00,00	545 Y7	DATA 3A,3E,43,F9,00,00,3A,98,20,3C
378 If	DATA 3A,3E,20,79,00,00,3A,62,43,F9	462 5G	DATA 3A,6A,30,3C,00,00,32,3C,00,55	546 mp	DATA 00,00,00,10,4E,AE,FE,AA,43,F9
379 oE	DATA 00,00,32,6E,30,3C,00,20,4E,AE	463 Ca	DATA 34,3C,00,35,36,3C,00,61,4E,AE	547 ze	DATA 00,00,3A,98,42,80,42,81,34,3C
380 mB	DATA FF,40,2C,79,00,00,3A,42,24,3C	464 GP	DATA FE,CE,22,79,00,00,3A,6A,30,3C	548 qD	DATA 00,1F,36,3C,00,1F,4E,AE,FE,CE
381 vR	DATA 00,00,03,ED,22,3C,00,00,2E,EA	465 Hk	DATA 00,59,32,3C,00,55,34,3C,00,85	549 hY	DATA 4E,B9,00,00,27,A6,60,00,F2,88
382 7m	DATA 4E,AE,FF,E2,23,00,00,00,3A,86	466 dJ	DATA 36,3C,00,61,4E,AE,FE,CE,22,79	550 Pm	DATA 20,79,00,00,3B,10,61,00,00,28
383 N1	DATA 67,00,01,06,22,00,26,3C,00,00	467 b5	DATA 00,00,3A,6A,30,3C,00,A9,32,3C	551 bE	DATA 20,79,00,00,3B,14,61,00,00,1E
384 hb	DATA 00,04,24,3C,00,00,00,00,4E,AE	468 7A	DATA 00,55,34,3C,00,DF,36,3C,00,61	552 Rr	DATA 20,79,00,00,3B,18,61,00,00,14
385 CJ	DATA FF,D6,67,00,00,CC,0C,B9,41,4F	469 b1	DATA 4E,AE,FE,CE,22,79,00,00,3A,6A	553 y1	DATA 20,79,00,00,3B,1C,61,00,00,0A
386 kJ	DATA 45,20,00,00,00,00,66,00,00,F6	470 dX	DATA 30,3C,00,00,32,3C,00,61,34,3C	554 ab	DATA 61,00,12,D8,60,00,F2,58,20,3C
387 k5	DATA 22,39,00,00,3A,86,26,3C,00,00	471 m0	DATA 00,DF,36,3C,00,00,72,4E,AE,FE,CE	555 k8	DATA 00,00,00,1F,22,10,24,10,48,42
388 pv	DATA 00,02,24,3C,00,00,00,04,4E,AE	472 PM	DATA 22,79,00,00,3A,6A,30,3C,00,00	556 xg	DATA C2,BC,00,00,FF,FF,C4,BC,00,00
389 cI	DATA FF,D6,67,00,00,A4,20,39,00,00	473 1a	DATA 32,3C,00,73,34,3C,00,1D,36,3C	557 I9	DATA FF,FF,42,83,42,84,2E,3C,00,00
390 sp	DATA 2E,62,D0,79,00,00,00,04,B0,B9	474 EK	DATA 00,7F,4E,AE,FE,CE,22,79,00,00	558 zR	DATA 00,0F,0F,01,67,00,00,10,46,47
391 xu	DATA 00,04,3A,78,62,00,00,8E,22,39	475 tN	DATA 3A,6A,30,3C,00,B1,32,3C,00,73	559 La	DATA CE,7C,00,0F,0F,C3,46,47,CE,7C
392 Iy	DATA 00,00,3A,86,26,3C,00,00,00,20	476 Ov	DATA 34,3C,00,DF,36,3C,00,7F,4E,AE	560 Hh	DATA 00,0F,51,CF,FF,EA,2E,3C,00,00
393 Sr	DATA 24,3C,00,00,32,8E,4E,AE,FF,D6	477 Te	DATA FE,CE,22,79,00,00,3A,6A,30,3C	561 4X	DATA 00,0F,0F,02,67,00,00,10,46,47
394 k3	DATA 67,00,00,74,42,82,24,3E,00,00	478 Fz	DATA 00,00,32,3C,00,7F,34,3C,00,DF	562 Rh	DATA CE,7C,00,0A,0F,C4,46,47,CE,7C
395 Ad	DATA 2E,62,94,BC,00,00,00,01,C4,FC	479 Y2	DATA 36,3C,00,DF,4E,AE,FE,CE,4B,F9	563 3s	DATA 00,0F,51,CF,FF,EA,48,43,86,84
396 mo	DATA 02,80,D4,B9,00,00,3A,4A,42,83	480 NB	DATA 00,00,3B,A5,3A,3C,00,14,3C,3C	564 oK	DATA 20,C3,51,C8,FF,A6,4E,75,20,79
397 51	DATA 36,39,00,00,00,04,C6,FC,02,80	481 NI	DATA 00,1F,36,3C,00,4F,4E,B9,00,00	565 yU	DATA 00,00,3B,10,61,00,00,28,20,79
398 xy	DATA 22,39,00,00,3A,86,4E,AE,FF,D6	482 bG	DATA 04,18,4B,F9,00,00,3B,B9,3A,3C	566 u6	DATA 00,00,3B,14,61,00,00,1E,20,79
399 iK	DATA 67,00,00,42,4E,B9,00,00,2A,F2	483 Gc	DATA 00,04,3C,3C,00,5F,3E,3C,00,5D	567 xw	DATA 00,00,3B,18,61,00,00,14,20,79
400 d1	DATA 2C,79,00,00,3A,3E,20,79,00,00	484 j1	DATA 4E,B9,00,00,04,18,4B,F9,00,00	568 MD	DATA 00,00,3B,1C,61,00,00,0A,61,00
401 ys	DATA 3A,62,43,F9,00,00,32,6E,30,3C	485 yS	DATA 3B,96,3A,3C,00,09,3C,00,1F	569 f8	DATA 12,44,60,00,F1,22,10,24,10,48,42
402 eV	DATA 00,20,4E,AE,FF,40,4E,B9,00,00	486 OQ	DATA 3E,3C,00,6B,4E,B9,00,00,04,18	570 tN	DATA 00,00,00,80,20,3C,00,00,00,0F
403 x5	DATA 27,A6,4E,B9,00,00,05,C2,2C,79	487 Fs	DATA 4B,F9,00,00,3B,9F,3A,3C,00,06	571 Ee	DATA 22,10,24,21,22,81,20,C2,51,C8
404 sn	DATA 00,00,3A,42,22,39,00,00,3A,86	488 4J	DATA 3C,3C,00,58,3E,3C,00,91,4E,B9	572 Tg	DATA FF,F6,4E,75,20,79,00,00,3B,10
405 cI	DATA 4E,AE,FF,DC,60,00,F8,2A,20,79	489 7E	DATA 00,00,04,18,30,3C,00,02,38,3C	573 zp	DATA 61,00,00,28,20,79,00,00,3B,14
406 i4	DATA 00,00,2D,B4,2C,79,00,00,3A,46	490 wJ	DATA 00,54,3A,3C,00,87,3C,3C,00,8C	574 ep	DATA 61,00,00,1E,20,79,00,00,3B,18
407 pn	DATA 4E,AE,FF,A0,2C,79,00,00,3A,42	491 Ra	DATA 3E,3C,00,96,4E,B9,00,00,04,88	575 05	DATA 61,00,00,14,20,79,00,00,3B,1C
408 A0	DATA 22,39,00,00,00,3A,86,4E,AE,FF,DC	492 mE	DATA 30,39,00,00,30,02,08,00,00,07	576 th	DATA 61,00,00,0A,61,00,11,F8,60,00
409 gS	DATA 60,00,F8,06,20,79,00,00,2D,B4	493 Qq	DATA 66,00,00,36,08,39,00,06,00,BF	577 Rd	DATA F1,78,43,F9,00,00,00,00,24,48
410 Ud	DATA 2C,79,00,00,3A,46,4E,AE,FF,A0	494 Jo	DATA E0,01,66,EB,22,79,00,00,3A,56	578 3S	DATA 2C,3C,00,00,00,1F,2E,3C,00,1F
411 10	DATA 60,00,F7,F2,0C,B9,46,4F,52,4D	495 L5	DATA 3A,29,00,0E,3C,29,00,0C,BA,7C	579 Cy	DATA 00,1F,42,83,20,4A,20,18,00,00
412 6F	DATA 00,00,00,00,66,BC,22,39,00,00	496 oR	DATA 00,54,65,D4,BC,7C,00,87,65,CE	580 Ae	DATA 67,00,00,10,46,47,CE,7C,00,1F
413 VQ	DATA 3A,86,26,3C,00,00,00,04,24,3C	497 gT	DATA BA,7C,00,8C,62,C8,BC,7C,00,96	581 V2	DATA 0F,C3,46,47,CE,7C,00,1F,51,CF
414 gN	DATA 00,00,00,00,4E,AE,FF,D6,67,A4	498 KZ	DATA 62,C2,60,00,00,0E,30,39,00,00	582 qw	DATA FF,E8,22,C3,51,CE,FF,D8,43,F9
415 GB	DATA 0C,B9,00,00,04,00,00,00,00,00	499 5N	DATA 30,02,08,00,00,07,66,F4,0C,B9	583 re	DATA 00,00,00,00,20,4A,2C,3C,00,00
416 Bk	DATA 64,98,22,39,00,00,3A,86,26,3C	500 u1	DATA 00,00,00,00,00,00,2F,86,66,00	584 6S	DATA 00,1F,20,D9,51,CE,FF,FC,4E,75
417 SE	DATA 00,00,04,00,24,3C,00,00,00,00	501 12	DATA 00,0C,23,FC,00,00,00,01,00,00	585 FI	DATA 20,79,00,00,3A,62,30,3C,00,1F
418 o4	DATA 4E,AE,FF,D6,41,F9,00,00,00,00	502 aF	DATA 2F,86,2C,79,00,00,3A,46,20,79	586 oX	DATA 32,3C,00,0D,34,01,36,01,2C,79
419 iY	DATA 20,18,B0,BC,49,4C,42,4D,66,00	503 50	DATA 00,00,3A,56,4E,AE,FF,B8,23,FC	587 Ob	DATA 00,00,3A,3E,4E,AE,FE,E0,20,79
420 eu	DATA FF,72,20,18,B0,BC,42,4D,48,44	504 YO	DATA 00,00,00,00,00,00,2D,A8,2C,79	588 4F	DATA 00,00,3A,62,30,3C,00,1E,32,3C
421 Iw	DATA 67,00,00,3E,B1,FC,00,00,04,00	505 X1	DATA 00,00,3A,3E,20,79,00,00,3A,62	589 ea	DATA 00,03,34,01,36,01,4E,AE,FE,E0
422 Sn	DATA 66,EC,60,00,00,FF,5A,20,18,B0,BC	506 OU	DATA 43,F9,00,00,00,32,6E,30,3C,00,20	590 fm	DATA 2C,79,00,00,3A,46,23,FC,00,00
423 iJ	DATA 43,4D,41,50,67,00,00,50,B1,FC	507 9v	DATA 4E,AE,FF,40,0C,B9,00,00,00,00	591 T2	DATA 31,5A,00,00,2D,A8,41,F9,00,00
424 Jb	DATA 00,00,04,00,66,EC,60,00,FF,42	508 lK	DATA 00,00,2F,86,66,00,00,0C,23,FC	592 cb	DATA 2D,96,4E,AE,FF,34,23,0C,00,00
425 xS	DATA 20,18,B0,BC,42,4F,44,59,67,00	509 oF	DATA 00,00,00,01,00,00,2F,86,0C,B9	593 fL	DATA 3A,56,20,40,23,EB,00,32,00,00
426 JH	DATA 00,6A,B1,FC,00,00,04,00,66,EC	510 RO	DATA 00,00,00,00,00,00,2F,E2,66,00	594 KM	DATA 3A,6A,23,EB,00,32,00,00,3A,6E
427 du	DATA 60,00,FF,2A,D1,FC,00,00,00,04	511 BC	DATA 00,0C,23,FC,00,00,00,01,00,00	595 dE	DATA 22,68,00,32,30,3C,00,01,2C,79
428 zz	DATA 20,18,B0,BC,00,20,00,20,66,00	512 Cy	DATA 2F,E2,20,39,00,00,2F,86,22,39	596 Vx	DATA 00,00,3A,3E,4E,AE,FE,AA,22,79
429 OE	DATA FF,18,D1,FC,00,00,00,04,10,18	513 UT	DATA 00,00,2F,E2,82,80,6C,00,00,0E	597 81	DATA 00,00,3A,6A,30,3C,00,01,4E,AE
430 w1	DATA B0,3C,00,04,66,00,FF,08,D1,FC	514 xI	DATA 23,C1,00,00,2F,86,23,C0,00,00	598 cp	DATA FE,A4,22,79,00,00,3A,6A,30,3C
431 Hy	DATA 00,00,00,03,60,A6,D1,FC,00,00	515 Ng	DATA 2F,E2,20,39,00,00,3A,78,B0,B9	599 uB	DATA 00,00,32,3C,00,00,34,3C,00,DF
432 G1	DATA 00,04,43,F9,00,00,32,8E,20,3C	516 40	DATA 00,00,2F,E2,65,00,01,04,2C,79	600 na	DATA 36,3C,00,54,4E,AE,FE,CE,22,79
433 Ow	DATA 00,00,00,0F,42,41,42,42,12,18	517 eB	DATA 00,00,3A,42,24,3C,00,00,03,EE	601 MQ	DATA 00,00,3A,6A,30,3C,00,00,32,3C
434 34	DATA E9,49,84,41,12,42,41,12,18,84,41	518 oZ	DATA 22,3C,00,00,2E,EA,4E,AE,FF,E2	602 uP	DATA 00,55,34,3C,00,35,36,3C,00,61
435 n4	DATA 42,41,12,18,EB,49,84,41,32,C2	519 4J	DATA 23,C0,00,00,3A,86,67,00,00,E4	603 lV	DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
436 1m	DATA 51,C8,FF,E4,60,8C,D1,FC,00,00	520 I7	DATA 23,FC,41,4F,45,20,00,00,00,00	604 mY	DATA 30,3C,00,59,32,3C,00,55,34,3C
437 16	DATA 00,04,20,39,00,00,2E,62,90,BC	521 vn	DATA 22,00,24,3C,00,00,00,00,26,3C	605 p9	DATA 00,85,36,3C,00,61,4E,AE,FE,CE
438 Mk	DATA 00,00,00,01,C0,FC,02,80,D0,B9	522 k4	DATA 00,00,00,04,4E,AE,FF,D0,67,00	606 Ol	DATA 22,79,00,00,3A,6A,30,3C,00,A9
439 4f	DATA 00,00,3A,4A,22,40,24,40,26,40	523 g2	DATA 00,A0,20,39,00,00,2F,E2,90,B9	607 qA	DATA 32,3C,00,55,34,3C,00,DF,36,3C
440 Pn	DATA 28,40,D5,FC,00,00,00,80,D7,FC	524 3C	DATA 00,00,2F,86,D0,7C,00,01,33,00	608 H1	DATA 00,61,4E,AE,FE,CE,22,79,00,00
441 Nv	DATA 00,00,01,00,D9,FC,00,00,01,80	525 8x	DATA 00,00,00,00,22,39,00,00,3A,86	609 CK	DATA 3A,6A,30,3C,00,00,32,3C,00,61
442 Th	DATA 20,3C,00,00,00,1F,22,D8,24,D8	526 w7	DATA 24,3C,00,00,00,00,26,3C,00,00	610 Dt	DATA 34,3C,00,DF,36,3C,00,68,4E,AE
443 21	DATA 26,D8,28,D8,51,C8,FF,F6,60,00	527 1a	DATA 00,02,4E,AE,FF,D0,67,00,00,70	611 dm	DATA FE,CE,22,79,00,00,3A,6A,30,3C
444 Xh	DATA FE,42,2C,79,00,00,3A,46,20,79	528 pI	DATA 22,39,00,00,3A,86,24,3C,00,00	612 rx	DATA 00,00,32,3C,00,69,34,3C,00,79
445 H8	DATA 00,00,3A,56,4E,AE,FF,B8,20,79	529 qp	DATA 32,8E,26,3C,00,00,00,20,4E,AE	613 Eu	DATA 36,3C,00,75,4E,AE,FE,CE,22,79
446 q2	DATA 00,00,3A,62,30,3C,00,1F,32,3C	530 w1	DATA FF,D0,67,00,00,56,4E,B9,00,00	614 Ip	DATA 00,00,3A,6A,30,3C,00,9D,32,3C
447 EY	DATA 00,0D,34,01,36,01,2C,79,00,00	531 Mw	DATA 2B,20,24,39,00,00,2F,86,94,BC	615 LE	DATA 00,69,34,3C,00,DF,36,3C,00,75
448 AP	DATA 3A,3E,4E,AE,FE,E0,20,79,00,00	532 Kq	DATA 00,00,00,01,C4,FC,02,80,D4,B9	616 y8	DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
449 zN	DATA 3A,62,30,3C,00,1E,32,3C,00,03	533 Lr	DATA 00,00,3A,4A,26,39,00,00,2F,E2	617 Ji	DATA 30,3C,00,00,32,3C,00,75,34,3C
450 Xn	DATA 34,01,36,01,4E,AE,FE,E0,20,79	534 Gc	DATA 96,B9,00,00,2F,86,D6,BC,00,00	618 76	DATA 00,DF,36,3C,00,00,DF,4E,AE,FE,CE
451 tF	DATA 00,00,3A,46,23,FC,00,00,2F,3E	535 H1	DATA 00,01,C6,FC,02,80,22,39,00,00	619 Te	DATA 4B,F9,00,00,3C,4F,3A,3C,00,13
452 yd	DATA 00,00,2D,A8,41,F9,00,00,2D,96	536 Dy	DATA 3A,86,4E,AE,FF,D0,67,00,00,16	620 rt	DATA 3C,3C,00,23,3E,3C,00,4F,4E,AE
453 OC	DATA 4E,AE,FF,34,23,0C,00,00,3A,56	537 Wg	DATA 4E,B9,00,00,2A,F2,22,39,00,00	621 F1	DATA 00,00,04,18,4B,F9,00,00,3B,B9
454 5p	DATA 20,40,23,EB,00,32,00,00,3A,6A	538 2V	DATA 3A,86,4E,AE,FF,DC,60,00,F2,F6	622 GN	DATA 3A,3C,00,04,3C,3C,00,5F,3E,3C
455 C1	DATA 23,EB,00,32,00,00,3A,6E,22,68	539 dE	DATA 20,79,00,00,2D,B4,2C,79,00,00	623 2t	DATA 00,5D,4E,B9,00,00,04,18,4B,F9
456 oE	DATA 00,32,30,3C,00,01,2C,79,00,00	540 41	DATA 3A,46,4E,AE,FF,A0,2C,79,00,00	624 gb	DATA 00,00,3C,62,3A,3C,00,06,3C,3C
457 CW	DATA 3A,3E,4E,AE,FE,AA,22,79,00,00	541 Ut	DATA 3A,42,22,39,00,00,3A,86,4E,AE	625 wB	DATA 00,3F,3E,3C,00,71,4E,B9,00,00

626 K3 DATA 04,18,4B,F9,00,00,3B,9F,3A,3C
627 PA DATA 00,06,3C,3C,00,58,3E,3C,00,91
628 78 DATA 4E,B9,00,00,04,18,30,3C,00,02
629 NU DATA 38,3C,00,54,3A,3C,00,87,3C,3C
630 RU DATA 00,8C,3E,3C,00,96,4E,B9,00,00
631 ex DATA 04,88,30,39,00,00,32,1E,08,00
632 vq DATA 00,07,66,00,00,36,08,39,00,06
633 3M DATA 00,BF,E0,01,66,E8,22,79,00,00
634 AQ DATA 3A,56,3A,29,00,0E,3C,29,00,0C
635 Ey DATA BA,7C,00,54,65,D4,BC,7C,00,87
636 9M DATA 65,CE,BA,7C,00,8C,62,C8,BC,7C
637 rF DATA 00,96,62,C2,60,00,00,0E,3C,39
638 xw DATA 00,00,32,1E,08,00,00,07,66,F4
639 nh DATA 2C,79,00,00,3A,46,20,79,00,00
640 OY DATA 3A,56,4E,AE,FF,B8,23,FC,00,00
641 w1 DATA 00,00,00,00,2D,A8,2C,79,00,00
642 Yv DATA 3A,3E,20,79,00,00,3A,62,43,F9
643 4U DATA 00,00,32,6E,30,3C,00,20,4E,AE
644 k2 DATA FF,40,4A,B9,00,00,31,A2,62,00
645 LF DATA 00,0C,23,FC,00,00,00,01,00,00
646 ry DATA 31,A2,4A,B9,00,00,31,FE,62,00
647 NO DATA 00,0C,23,FC,00,00,00,01,00,00
648 B1 DATA 31,FE,4A,B9,00,00,32,5A,63,00
649 JO DATA 01,00,20,39,00,00,31,A2,D0,B9
650 x0 DATA 00,00,32,5A,90,BC,00,00,00,01
651 Jm DATA B0,B9,00,00,3A,78,62,00,00,EE
652 3e DATA 22,39,00,00,31,FE,D2,B9,00,00
653 SQ DATA 32,5A,92,BC,00,00,00,01,B2,B9
654 hR DATA 00,00,3A,78,62,00,00,D2,B2,80
655 c8 DATA 67,00,00,C2,B0,B9,00,00,31,FE
656 aR DATA 62,00,00,4C,20,39,00,00,32,5A
657 dz DATA C0,FC,00,A0,90,BC,00,00,00,01
658 pb DATA 22,39,00,00,31,A2,92,BC,00,00
659 CP DATA 00,01,C2,FC,02,80,D2,B9,00,00
660 JQ DATA 3A,4A,20,41,22,39,00,00,31,FE
661 zV DATA 92,BC,00,00,00,01,C2,FC,02,80
662 40 DATA D2,B9,00,00,3A,4A,22,41,22,D8
663 av DATA 51,C8,FF,FC,60,00,00,6E,20,39
664 Ff DATA 00,00,31,A2,B0,B9,00,00,31,FE
665 Pk DATA 62,A8,20,39,00,00,32,5A,01,FC
666 95 DATA 00,A0,90,BC,00,00,00,00,C2,39
667 CR DATA 00,00,31,A2,D2,B9,00,00,32,5A
668 6c DATA 92,BC,00,00,00,01,C2,FC,02,80
669 yA DATA D2,B9,00,00,3A,4A,20,41,22,39
670 LZ DATA 00,00,31,FE,D2,B9,00,00,32,5A
671 9f DATA 92,BC,00,00,00,01,C2,FC,02,80
672 xJ DATA D2,B9,00,00,3A,4A,22,41,D1,FC
673 Tt DATA 00,00,00,04,D3,FC,00,00,00,04
674 HA DATA 23,20,51,C8,FF,FC,4E,B9,00,00
675 yM DATA 27,A6,60,00,ED,A0,2C,79,00,00
676 00 DATA 3A,46,20,79,00,00,2D,84,4E,AE
677 ZV DATA FF,A0,60,00,ED,00,41,F9,00,00
678 t6 DATA 2D,96,2C,79,00,00,3A,46,4E,AE
679 FW DATA FF,34,4A,80,67,00,ED,76,23,C0
680 UK DATA 00,00,3A,56,20,40,23,E8,00,32
681 WW DATA 00,00,3A,6A,22,68,00,32,30,3C
682 dt DATA 00,01,2C,79,00,00,3A,3E,4E,AE
683 Fc DATA FE,AA,22,79,00,00,3A,6A,30,3C
684 8A DATA 00,01,4E,AE,FE,A4,22,79,00,00
685 Tt DATA 3A,6A,42,40,42,41,34,3C,00,DF
686 Da DATA 36,3C,00,DF,4E,AE,FE,CE,23,F9
687 6U DATA 00,00,3A,6A,00,00,3A,6E,4B,F9
688 ER DATA 00,00,3B,BD,3A,3C,00,0C,3C,3C
689 or DATA 00,40,3E,3C,00,50,4E,B9,00,00
690 Og DATA 04,18,4B,F9,00,00,3B,C9,3A,3C
691 pv DATA 00,0B,3C,3C,00,44,3E,3C,00,5A
692 54 DATA 4E,B9,00,00,04,18,4B,F9,00,00
693 K4 DATA 3B,D4,3A,3C,00,0F,3C,3C,00,34
694 4L DATA 3E,3C,00,69,4E,B9,00,00,04,18
695 M8 DATA 4B,F9,00,00,3B,E3,3A,3C,00,0F
696 8r DATA 3C,3C,00,34,3E,3C,00,73,4E,B9
697 At DATA 00,00,04,18,4B,F9,00,00,3B,F2
698 93 DATA 3A,3C,00,11,3C,3C,00,2C,3E,3C
699 1p DATA 00,82,4E,B9,00,00,04,18,4B,F9
700 m0 DATA 00,00,3B,9F,3A,3C,00,06,3C,3C
701 Hz DATA 00,58,3E,3C,00,96,4E,B9,00,00
702 Yb DATA 04,18,30,3C,00,02,38,3C,00,54
703 me DATA 3A,3C,00,8C,3C,3C,00,04,3E,3C
704 SP DATA 00,9B,4E,B9,00,00,00,88,08,39
705 1V DATA 00,06,00,BF,E0,01,66,FE,22,79
706 Uy DATA 00,00,3A,56,3A,29,00,0E,3C,29
707 Je DATA 00,0C,BA,7C,00,54,65,E2,BC,7C
708 mc DATA 00,8C,65,DC,BA,7C,00,8C,62,D6
709 us DATA BC,7C,00,9B,62,D0,2C,79,00,00

710 1W DATA 3A,46,20,79,00,00,3A,56,4E,AE
711 CM DATA FF,B8,60,00,EC,38,41,F9,00,00
712 3k DATA 2D,96,23,FC,00,00,30,A2,00,00
713 IV DATA 2D,A8,2C,79,00,00,3A,46,4E,AE
714 1h DATA FF,34,23,C0,00,00,3A,56,20,40
715 Bq DATA 23,E8,00,32,00,00,3A,6A,23,E8
716 HQ DATA 00,32,00,00,3A,6E,22,68,00,32
717 p2 DATA 30,3C,00,01,2C,79,00,00,3A,3E
718 IW DATA 4E,AE,FE,AA,22,79,00,00,3A,6A
719 aD DATA 30,3C,00,01,4E,AE,FE,A4,20,79
720 GS DATA 00,00,3A,62,30,3C,00,1F,32,3C
721 ey DATA 00,0D,34,01,36,01,2C,79,00,00
722 ap DATA 3A,3E,4E,AE,FE,E0,20,79,00,00
723 Pn DATA 3A,62,30,3C,00,1E,32,3C,00,03
724 PO DATA 34,01,36,01,4E,AE,FE,E0,22,79
725 MQ DATA 00,00,3A,6A,30,3C,00,00,32,3C
726 xQ DATA 00,00,3A,3C,00,0F,3B,3C,00,5E
727 1v DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
728 6K DATA 30,3C,00,00,32,3C,00,5F,34,3C
729 Fl DATA 00,35,36,3C,00,6B,4E,AE,FE,CE
730 en DATA 22,79,00,00,3A,6A,30,3C,00,59
731 1q DATA 32,3C,00,5F,34,3C,00,85,36,3C
732 vQ DATA 00,6B,4E,AE,FE,CE,22,79,00,00
733 og DATA 3A,6A,30,3C,00,A9,32,3C,00,5F
734 rh DATA 34,3C,00,DF,36,3C,00,6B,4E,AE
735 dm DATA FE,CE,22,79,00,00,3A,6A,30,3C
736 Bq DATA 00,00,32,3C,00,6B,34,3C,00,DF
737 1C DATA 36,3C,00,DF,4E,AE,FE,CE,4B,F9
738 RU DATA 00,00,3C,1A,3A,3C,00,14,3C,3C
739 wF DATA 00,1F,3E,3C,00,59,4E,B9,00,00
740 QJ DATA 04,18,4B,F9,00,00,3B,B9,3A,3C
741 MV DATA 00,04,3C,3C,00,5F,3E,3C,00,66
742 ts DATA 4E,B9,00,00,04,18,4B,F9,00,00
743 fu DATA 3B,9F,3A,3C,00,06,3C,3C,00,58
744 K1 DATA 3E,3C,00,7D,4E,B9,00,00,04,18
745 BS DATA 30,3C,00,02,38,3C,00,54,3A,3C
746 fx DATA 00,73,3C,3C,00,8C,3E,3C,00,82
747 TW DATA 4E,B9,00,00,04,88,08,39,00,06
748 q8 DATA 00,BF,E0,01,66,F6,22,79,00,00
749 1H DATA 3A,56,3A,29,00,0E,3C,29,00,0C
750 dH DATA BA,7C,00,54,65,E2,BC,7C,00,73
751 o9 DATA 65,DC,BA,7C,00,8C,62,D6,BC,7C
752 EA DATA 00,82,62,D0,2C,79,00,00,3A,46
753 qq DATA 20,79,00,00,3A,56,4E,AE,FF,B8
754 q7 DATA 0C,B9,00,00,00,00,30,3C,00,EA
755 Ms DATA 66,00,00,0C,23,FC,00,00,00,01
756 WI DATA 00,00,30,EA,20,39,00,00,3A,78
757 1D DATA B0,B9,00,00,30,EA,62,00,00,0C
758 HN DATA 23,F9,00,00,3A,78,00,00,30,EA
759 Yo DATA 0C,B9,00,00,00,00,00,00,31,46
760 Rx DATA 66,00,00,0C,23,FC,00,00,00,01
761 FN DATA 00,00,31,46,20,39,00,00,3A,78
762 6r DATA B0,B9,00,00,31,46,62,00,00,0C
763 P4 DATA 23,F9,00,00,3A,78,00,00,31,46
764 Hn DATA 20,39,00,00,3A,EA,22,39,00,00
765 pN DATA 31,46,B2,80,64,00,00,0E,23,C1
766 8J DATA 00,00,30,EA,23,C0,00,00,31,46
767 yJ DATA 23,F9,00,00,30,EA,00,00,3A,8E
768 HU DATA 41,F9,00,00,2D,96,23,FC,00,00
769 05 DATA 00,00,00,00,2D,A8,2C,79,00,00
770 Yq DATA 3A,46,4E,AE,FF,34,23,C0,00,00
771 LD DATA 3A,56,20,40,23,E8,00,32,00,00
772 CE DATA 3A,6A,23,E8,00,32,00,00,3A,6E
773 nJ DATA 22,68,00,32,22,79,00,00,3A,6A
774 kx DATA 30,3C,00,01,2C,79,00,00,3A,3E
775 qJ DATA 4E,AE,FE,AA,20,79,00,00,3A,62
776 kq DATA 43,F9,00,00,32,6E,30,3C,00,20
777 VJ DATA 4E,AE,FF,40,22,79,00,00,3A,6A
778 bg DATA 30,3C,00,01,4E,AE,FE,A4,22,79
779 EI DATA 00,00,3A,6A,30,3C,00,00,32,3C
780 r1 DATA 00,00,34,3C,00,DF,36,3C,00,DF
781 zd DATA 4E,AE,FE,CE,2C,79,00,00,3A,46
782 xQ DATA 20,79,00,00,3A,6A,43,F9,00,00
783 12 DATA 35,9E,30,3C,00,58,32,3C,00,57
784 BM DATA 4E,AE,FF,8E,2C,79,00,00,3A,3E
785 9v DATA 4B,F9,00,00,3C,2E,3A,3C,00,0F
786 fe DATA 3C,3C,00,34,3E,3C,00,4F,4E,B9
787 OE DATA 00,00,04,18,4B,F9,00,00,3B,9F
788 61 DATA 3A,3C,00,06,3C,3C,00,58,3E,3C
789 s1 DATA 00,7D,4E,B9,00,00,04,18,4B,F9
790 S0 DATA 00,00,3C,3D,3A,3C,00,03,3C,3C
791 s5 DATA 00,63,3E,3C,00,5D,4E,B9,00,00
792 03 DATA 04,18,30,3C,00,02,38,3C,00,54
793 eG DATA 3A,3C,00,73,3C,3C,00,8C,3E,3C

794 XL DATA 00,82,4E,B9,00,00,04,88,30,3C
795 EH DATA 00,10,22,79,00,00,3A,6A,4E,AE
796 4R DATA FE,AA,22,79,00,00,3A,6A,30,3C
797 wJ DATA 00,5F,32,3C,00,14,34,3C,00,7E
798 LF DATA 36,3C,00,33,4E,AE,FE,CE,30,3C
799 Ob DATA 00,02,38,3C,00,5E,3A,3C,00,13
800 eM DATA 3C,3C,00,7F,3E,3C,00,34,4E,B9
801 tZ DATA 00,00,04,88,30,3C,00,03,38,3C
802 WM DATA 00,5D,3A,3C,00,12,3C,3C,00,80
803 rN DATA 3E,3C,00,35,4E,B9,00,00,04,88
804 A1 DATA 30,3C,00,04,38,3C,00,5C,3A,3C
805 45 DATA 00,11,3C,3C,00,81,3E,3C,00,36
806 XZ DATA 4E,B9,00,00,04,88,23,F9,00,00
807 A7 DATA 3B,10,00,00,00,00,23,F9,00,00
808 1W DATA 3B,14,00,00,00,04,23,F9,00,00
809 Mv DATA 3B,18,00,00,00,08,23,F9,00,00
810 y0 DATA 3B,1C,00,00,00,00,23,F9,00,00
811 LK DATA 3B,20,00,00,00,10,23,F9,00,00
812 1N DATA 3A,92,00,00,3A,8A,2A,7C,00,DF
813 45 DATA F0,00,23,F8,00,6C,00,00,27,68
814 FM DATA 21,FC,00,00,26,AC,00,6C,0C,79
815 61 DATA 00,01,00,00,3A,96,66,00,00,54
816 JT DATA 20,39,00,DF,F0,04,C0,BC,00,01
817 tF DATA FF,00,B0,BC,00,01,00,00,66,EC
818 Mh DATA 2C,79,00,00,3A,3E,41,F9,00,00
819 4G DATA 3A,98,22,79,00,00,3A,6A,42,80
820 Ec DATA 42,81,24,3C,00,00,00,5F,26,3C
821 OC DATA 00,00,00,14,28,3C,00,00,20,20
822 fF DATA 2A,3C,00,00,00,20,2C,3C,00,00
823 B1 DATA 00,C0,4E,AE,FD,D8,42,79,00,00
824 J8 DATA 3A,96,08,39,00,00,00,BF,E0,01
825 hv DATA 66,98,22,79,00,00,3A,56,3A,29
826 1A DATA 00,0E,3C,29,00,0C,BA,7C,00,58
827 pR DATA 65,00,00,1E,BC,7C,00,57,65,00
828 sv DATA 00,16,BA,7C,00,5F,62,00,00,0E
829 xY DATA BC,7C,00,5E,62,00,00,06,60,00
830 Z6 DATA 0A,06,BA,7C,00,80,65,00,00,22
831 E1 DATA BC,7C,00,57,65,00,00,1A,BA,7C
832 AB DATA 00,87,62,00,00,12,BC,7C,00,5E
833 7S DATA 62,00,00,0A,60,00,0A,22,60,00
834 oI DATA FF,40,BA,7C,00,54,65,00,FF,38
835 GI DATA BC,7C,00,73,65,00,FF,30,BA,7C
836 ZN DATA 00,8C,62,00,FF,28,BC,7C,00,82
837 da DATA 62,00,FF,20,2A,7C,00,DF,F0,00
838 Vm DATA 21,F9,00,00,27,68,00,00,62,3F
839 0a DATA 00,00,00,00,00,00,3B,10,23,F9
840 P7 DATA 00,00,00,04,00,00,3B,14,23,F9
841 oe DATA 00,00,00,08,00,00,3B,18,23,F9
842 t5 DATA 00,00,00,0C,00,00,3B,1C,23,F9
843 1J DATA 00,00,00,10,00,00,3B,20,2C,79
844 sq DATA 00,00,3A,46,20,79,00,00,3A,56
845 t0 DATA 4E,AE,FF,B8,23,F9,00,00,3A,66
846 9Q DATA 00,00,3A,6E,23,FC,00,00,00,00
847 EM DATA 00,00,2D,A8,60,00,E6,E6,30,39
848 wb DATA 00,00,3A,76,B0,7C,00,01,66,00
849 53 DATA 00,06,60,00,00,26,D4,04,B9,00,00
850 yv DATA 02,80,00,00,3B,10,04,B9,00,00
851 BC DATA 02,80,00,00,3B,14,04,B9,00,00
852 OT DATA 02,80,00,00,3B,18,04,B9,00,00
853 wC DATA 02,80,00,00,3B,1C,04,B9,00,00
854 cQ DATA 02,80,00,00,3B,20,04,79,00,01
855 2L DATA 00,00,3A,76,4E,B9,00,00,2A,42
856 PX DATA 4E,B9,00,00,27,A6,60,00,E6,8A
857 13 DATA 30,39,00,00,3A,76,B0,B9,00,00
858 1E DATA 3A,78,66,00,00,06,60,00,E6,76
859 pA DATA 06,B9,00,00,02,80,00,00,3B,10
860 AZ DATA 06,B9,00,00,02,80,00,00,3B,14
861 Vy DATA 06,B9,00,00,02,80,00,00,3B,18
862 P3 DATA 06,B9,00,00,02,80,00,00,3B,1C
863 xJ DATA 06,B9,00,00,02,80,00,00,3B,20
864 Gq DATA 06,79,00,01,00,00,3A,76,4E,B9
865 Z6 DATA 00,00,2A,42,4E,B9,00,00,27,A6
866 1S DATA 60,00,E6,2C,42,80,30,39,00,00
867 hy DATA 3A,72,C0,FC,00,02,20,40,D1,FC
868 14 DATA 00,00,32,8E,30,10,C0,7C,0F,00
869 3U DATA E0,58,D0,7C,00,01,C0,7C,0F,00
870 GW DATA E1,58,02,50,00,FF,81,50,4E,B9
871 Jt DATA 00,00,27,6C,4E,B9,00,00,05,C2
872 NC DATA 60,00,E5,F0,42,80,30,39,00,00
873 n4 DATA 3A,72,C0,FC,00,02,20,40,D1,FC
874 rA DATA 00,00,32,8E,30,10,C0,7C,0F,00

Listing 1. (Fortsetzung)

875 cs DATA E0,58,90,7C,00,01,C0,7C,00,0F
 876 Mc DATA E1,58,02,50,00,BF,81,50,4E,B9
 877 Pz DATA 00,00,27,6C,4E,B9,00,00,05,C2
 878 DM DATA 60,00,E5,B4,42,80,30,39,00,00
 879 ta DATA 3A,72,C0,FC,00,02,20,40,D1,FC
 880 fy DATA 00,00,32,8E,30,10,C0,7C,00,F0
 881 dC DATA E8,58,D0,7C,00,01,C0,7C,00,0F
 882 uI DATA E9,58,02,50,0F,0F,81,50,4E,B9
 883 V5 DATA 00,00,27,6C,4E,B9,00,00,05,C2
 884 NI DATA 60,00,E5,78,42,80,30,39,00,00
 885 zG DATA 3A,72,C0,FC,00,02,20,40,D1,FC
 886 lA DATA 00,00,32,8E,30,10,C0,7C,00,F0
 887 Ca DATA E8,58,90,7C,00,01,C0,7C,00,0F
 888 00 DATA E9,58,02,50,0F,0F,81,50,4E,B9
 889 bn DATA 00,00,27,6C,4E,B9,00,00,05,C2
 890 Rn DATA 60,00,E5,3C,42,80,30,39,00,00
 891 5M DATA 3A,72,C0,FC,00,02,20,40,D1,FC
 892 DW DATA 00,00,32,8E,30,10,C0,7C,00,0F
 893 fL DATA D0,7C,00,01,C0,7C,00,0F,02,50
 894 vW DATA 0F,0F,81,50,4E,B9,00,00,27,6C
 895 lh DATA 4E,B9,00,00,05,C2,60,00,E5,04
 896 CD DATA 42,80,30,39,00,00,3A,72,C0,FC
 897 iv DATA 00,02,20,40,D1,FC,00,00,32,8E
 898 Jj DATA 30,10,C0,7C,00,0F,90,7C,00,01
 899 sJ DATA C0,7C,00,0F,02,50,0F,F0,81,50
 900 bf DATA 4E,B9,00,00,27,6C,4E,B9,00,00
 901 vc DATA 05,C2,60,00,E4,C2,60,00,00,00
 902 JD DATA 3A,46,41,F9,00,00,2D,C6,4E,AE
 903 kK DATA FF,34,23,C0,00,00,3A,56,20,40
 904 Bt DATA 23,E8,00,32,00,00,3A,6A,23,E8
 905 KT DATA 00,32,00,00,3A,6E,22,68,00,32
 906 s5 DATA 30,3C,00,01,2C,79,00,00,3A,3E
 907 yv DATA 4E,AE,FE,AA,20,79,00,00,3A,62
 908 nr DATA 30,3C,00,1F,32,00,00,3A,3C,01
 909 tC DATA 36,01,4E,AE,FE,E0,20,79,00,00
 910 Qp DATA 3A,62,30,3C,00,1E,32,3C,00,03
 911 QP DATA 3A,01,36,01,4E,AE,FE,E0,22,79
 912 Dn DATA 00,00,3A,6A,30,3C,00,01,4E,AE
 913 OH DATA FE,A4,22,79,00,00,3A,6A,42,40
 914 k8 DATA 42,41,34,3C,00,0F,36,3C,00,0C
 915 DM DATA 4E,AE,FE,CE,42,40,32,3C,00,0D
 916 KE DATA 34,3C,00,36,36,3C,00,18,22,79
 917 m4 DATA 00,00,3A,6A,4E,AE,FE,CE,30,3C
 918 Ig DATA 00,5A,32,3C,00,0D,3A,3C,00,8F
 919 CG DATA 36,3C,00,18,22,79,00,00,3A,6A
 920 TS DATA 4E,AE,FE,CE,42,40,32,3C,00,19
 921 AC DATA 34,3C,00,0F,36,3C,00,2C,22,79
 922 r9 DATA 00,00,3A,6A,4E,AE,FE,CE,30,3C
 923 YQ DATA 00,03,42,44,42,45,3C,3C,00,8F
 924 Ay DATA 3E,3C,00,2C,4E,B9,00,00,04,88
 925 Is DATA 4B,F9,00,00,3C,03,3A,3C,00,10
 926 fc DATA 3C,3C,00,08,3E,3C,00,08,4E,B9
 927 eU DATA 00,00,04,18,4B,F9,00,00,3B,9F
 928 kV DATA 3A,3C,00,06,3C,3C,00,30,3E,3C
 929 Ba DATA 00,25,4E,B9,00,00,04,18,30,3C
 930 Wt DATA 00,02,38,3C,00,2C,3A,3C,00,1B
 931 sY DATA 3C,3C,00,04,3E,3C,00,2A,4E,B9
 932 pv DATA 00,00,04,88,30,39,00,00,30,52
 933 FP DATA 08,00,00,07,66,00,00,36,08,39
 934 XI DATA 00,06,00,BF,E0,01,66,E8,22,79
 935 Bf DATA 00,00,3A,56,3A,29,00,0E,3C,29
 936 IE DATA 00,0C,BA,7C,00,2C,65,D4,BC,7C
 937 oR DATA 00,1B,65,CE,BA,7C,00,64,62,C8
 938 DO DATA BC,7C,00,2A,62,C2,60,00,00,0E
 939 9g DATA 30,39,00,00,30,52,08,00,00,07
 940 wI DATA 66,F4,2C,79,00,00,3A,46,20,79
 941 Ps DATA 00,00,3A,56,4E,AE,FF,B8,0C,B9
 942 xR DATA 00,00,00,00,00,00,00,8E,66,00
 943 9A DATA 00,0C,23,FC,00,00,00,01,00,00
 944 3a DATA 30,8E,20,39,00,00,3A,78,B0,B9
 945 DE DATA 00,00,30,8E,64,00,00,0C,23,F9
 946 XN DATA 00,00,3A,78,00,00,30,8E,20,39
 947 hW DATA 00,00,30,8E,33,C0,00,00,3A,76
 948 Kw DATA 90,BC,00,00,00,01,C0,FC,02,80
 949 AG DATA D0,B9,00,00,3A,4A,23,C0,00,00
 950 b2 DATA 3B,10,D0,BC,00,00,00,80,23,C0
 951 Bp DATA 00,00,3B,14,D0,BC,00,00,00,80
 952 sv DATA 23,C0,00,00,3B,18,D0,BC,00,00
 953 2n DATA 00,80,23,C0,00,00,3B,1C,D0,BC
 954 9P DATA 00,00,00,80,23,C0,00,00,3B,20
 955 aI DATA 2C,79,00,00,3A,3E,20,79,00,00
 956 vp DATA 3A,62,43,F9,00,00,32,6E,30,3C
 957 bS DATA 00,20,4E,AE,FF,04,4E,B9,00,00
 958 4W DATA 2A,42,4E,B9,00,00,27,A6,60,00

959 Jy DATA E2,8C,23,FC,00,00,00,64,00,00
 960 lK DATA 30,8E,20,3C,00,01,30,30,E1,88
 961 OT DATA 23,C0,00,00,30,96,23,C0,00,00
 962 V8 DATA 30,9C,2C,79,00,00,3A,46,41,F9
 963 lA DATA 00,00,2D,C6,4E,AE,FF,34,23,C0
 964 4u DATA 00,00,3A,56,20,40,23,E8,00,32
 965 ek DATA 00,00,3A,6A,23,E8,00,32,00,00
 966 yb DATA 3A,6E,22,68,00,32,30,3C,00,01
 967 V8 DATA 2C,79,00,00,3A,3E,4E,AE,FE,AA
 968 QT DATA 20,79,00,00,3A,62,30,3C,00,1F
 969 rQ DATA 32,3C,00,0D,34,01,36,01,4E,AE
 970 SI DATA FE,E0,20,79,00,00,3A,62,30,3C
 971 Mf DATA 00,1E,32,3C,00,03,34,01,36,01
 972 Ev DATA 4E,AE,FE,E0,22,79,00,00,3A,6A
 973 kP DATA 30,3C,00,01,4E,AE,FE,A4,22,79
 974 YQ DATA 00,00,3A,6A,42,40,42,41,34,3C
 975 7X DATA 00,8F,36,3C,00,0C,4E,AE,FE,CE
 976 7L DATA 42,40,32,3C,00,0D,34,3C,00,36
 977 8C DATA 36,3C,00,18,22,79,00,00,3A,6A
 978 tE DATA 4E,AE,FE,CE,30,3C,00,0A,32,3C
 979 G6 DATA 00,0D,34,3C,00,0F,36,3C,00,18
 980 FG DATA 22,79,00,00,3A,6A,4E,AE,FE,CE
 981 JI DATA 42,40,32,3C,00,0D,34,3C,00,36
 982 fQ DATA 36,3C,00,2C,22,79,00,00,3A,6A
 983 mW DATA 4E,AE,FE,CE,30,3C,00,03,42,44
 984 oF DATA 42,45,3C,3C,00,0F,3E,3C,00,2C
 985 28 DATA 4E,B9,00,00,04,88,4B,F9,00,00
 986 RO DATA 3C,40,3A,3C,00,0F,3C,3C,00,0C
 987 Bp DATA 3E,3C,00,08,4E,B9,00,00,04,18
 988 Kx DATA 4B,F9,00,00,00,3B,9F,3A,3C,00,06
 989 eE DATA 3C,3C,00,30,3E,3C,00,25,4E,B9
 990 CJ DATA 00,00,04,18,30,3C,00,02,38,3C
 991 CL DATA 00,2C,3A,3C,00,1B,3C,3C,00,64
 992 Cy DATA 3E,3C,00,2A,4E,B9,00,00,04,88
 993 lY DATA 30,39,00,00,30,52,08,00,00,07
 994 Vv DATA 66,00,00,36,08,39,00,06,00,BF
 995 OT DATA E0,01,66,E8,22,79,00,00,3A,56
 996 QA DATA 3A,29,00,0E,3C,29,00,0C,BA,7C
 997 T9 DATA 00,2C,65,D4,BC,7C,00,1B,65,CE
 998 WG DATA BA,7C,00,64,62,C8,BC,7C,00,2A
 999 Pe DATA 62,C2,60,00,00,0E,3C,30,00,0D
 1000 37 DATA 30,52,08,00,00,07,66,F4,2C,79
 1001 PN DATA 00,00,3A,46,20,79,00,00,3A,56
 1002 kK DATA 4E,AE,FF,B8,0C,B9,00,00,00,00
 1003 7K DATA 00,00,30,8E,66,00,00,0C,23,FC
 1004 er DATA 00,00,00,01,00,00,30,8E,23,F9
 1005 XI DATA 00,00,30,8E,00,00,3A,78,2C,79
 1006 eQ DATA 00,00,3A,3E,20,79,00,00,3A,62
 1007 TZ DATA 43,F9,00,00,32,6E,30,3C,00,20
 1008 lY DATA 4E,AE,FF,04,23,FC,00,00,00,01
 1009 JA DATA 00,00,30,8E,20,3C,00,00,30,30
 1010 9k DATA E1,88,23,C0,00,00,30,96,23,C0
 1011 Ge DATA 00,00,30,9C,4E,75,08,39,00,05
 1012 mU DATA 00,DF,0F,1F,66,00,00,06,60,00
 1013 jB DATA 00,AC,48,E7,FF,FF,0C,B9,00,00
 1014 Es DATA 00,00,00,00,3A,8A,67,00,00,10
 1015 fC DATA 04,B9,00,00,00,01,00,00,3A,8A
 1016 K4 DATA 60,00,00,88,20,39,00,00,3A,8E
 1017 R3 DATA 90,BC,00,00,00,01,C0,FC,02,80
 1018 HN DATA D0,B9,00,00,3A,4A,23,C0,00,00
 1019 i9 DATA 3B,10,D0,BC,00,00,00,80,23,C0
 1020 lW DATA 00,00,3B,14,D0,BC,00,00,00,80
 1021 z2 DATA 23,C0,00,00,3B,18,D0,BC,00,00
 1022 9u DATA 00,80,23,C0,00,00,3B,1C,D0,BC
 1023 GW DATA 00,00,00,80,23,C0,00,00,3B,20
 1024 Yf DATA 33,FC,00,01,00,00,3A,96,23,F9
 1025 Ce DATA 00,00,3A,92,00,00,3A,8A,20,39
 1026 Ub DATA 00,00,3A,8E,B0,B9,00,00,31,46
 1027 g4 DATA 66,00,00,10,23,F9,00,00,30,EA
 1028 yJ DATA 00,00,3A,8E,60,00,00,0C,06,B9
 1029 Kk DATA 00,00,00,01,00,00,3A,8E,4C,DF
 1030 lZ DATA 7F,FF,4E,F9,00,00,00,00,2C,79
 1031 ca DATA 00,00,3A,3E,42,80,42,81,42,82
 1032 oK DATA 42,83,30,39,00,00,3A,72,D0,7C
 1033 Qk DATA 00,10,32,10,34,10,36,10,C2,7C
 1034 HK DATA 0F,00,EO,59,C4,7C,00,FO,E8,5A
 1035 tI DATA C6,7C,00,0F,20,79,00,00,3A,62
 1036 3d DATA 4E,AE,FE,E0,4E,75,2C,79,00,00
 1037 sL DATA 3A,3E,41,F9,00,00,00,3A,98,22,79
 1038 YN DATA 00,00,3A,66,42,80,42,81,34,3C
 1039 qJ DATA 01,00,36,3C,00,08,38,3C,00,4E
 1040 UT DATA 3A,3C,00,20,3C,3C,00,0C,4E,AE
 1041 jQ DATA FD,D8,4E,B9,00,00,28,C0,22,79
 1042 lR DATA 00,00,3A,66,30,3C,00,02,4E,AE

1043 Vv DATA FE,AA,42,86,22,79,00,00,3A,66
 1044 W7 DATA 33,46,00,24,33,7C,00,00,00,26
 1045 87 DATA 30,06,32,3C,00,EO,4E,AE,FF,0A
 1046 cG DATA 22,79,00,00,3A,66,33,7C,00,00
 1047 gW DATA 00,24,33,46,00,26,30,3C,00,EO
 1048 9M DATA 32,06,4E,AE,FF,0A,DC,7C,00,07
 1049 w8 DATA BC,7C,00,E7,66,C2,22,79,00,00
 1050 yH DATA 3A,66,30,3C,00,04,4E,AE,FE,AA
 1051 9u DATA 22,79,00,00,3A,66,33,7C,00,70
 1052 hz DATA 00,24,33,7C,00,01,00,26,30,3C
 1053 Bx DATA 00,70,32,3C,00,DF,4E,AE,FF,0A
 1054 pU DATA 22,79,00,00,3A,66,33,7C,00,01
 1055 vJ DATA 00,24,33,7C,00,70,00,26,30,3C
 1056 vv DATA 00,DF,32,3C,00,70,4E,AE,FF,0A
 1057 lB DATA 2C,78,00,04,4E,AE,FF,7C,2C,3C
 1058 YC DATA 00,00,00,1F,2E,3C,00,00,00,1F
 1059 r3 DATA 20,07,22,06,4E,B9,00,00,2A,94
 1060 qV DATA 20,07,22,06,C0,FC,00,07,C2,FC
 1061 sY DATA 00,07,D0,BC,00,00,00,01,D2,BC
 1062 oz DATA 00,00,00,01,4E,B9,00,00,29,30
 1063 c5 DATA 51,CF,FF,D6,51,CE,FF,CC,2C,78
 1064 OA DATA 00,04,4E,AE,FF,76,4E,75,4E,AE
 1065 G4 DATA FE,38,2A,7C,00,0D,DF,00,20,7F
 1066 g5 DATA 00,00,3A,5A,22,28,00,08,4E,B9
 1067 UQ DATA 00,00,28,FC,22,28,00,0C,4E,B9
 1068 kO DATA 00,00,28,FC,22,28,00,10,4E,B9
 1069 bJ DATA 00,00,28,FC,22,28,00,14,4E,B9
 1070 D6 DATA 00,00,28,FC,4E,AE,FE,32,08,2D
 1071 9r DATA 00,0E,00,02,66,F8,2B,41,00,54
 1072 LH DATA 3B,7C,00,0C,00,66,3B,7C,FF,FF
 1073 ce DATA 00,44,3B,7C,FF,FF,00,46,42,6D
 1074 ot DATA 00,42,42,6D,00,74,3B,7C,01,F0
 1075 nJ DATA 00,40,3B,7C,38,0E,00,58,4E,75
 1076 sT DATA 36,00,C6,BC,00,00,00,0F,24,38
 1077 Ou DATA FC,00,00,00,E6,AA,C2,FC,00,28
 1078 aZ DATA E6,48,08,80,00,00,D2,80,76,28
 1079 Ye DATA 20,79,00,00,3A,5A,08,05,00,00
 1080 Le DATA 67,08,22,68,00,08,D3,C1,61,2C
 1081 NE DATA 08,05,00,01,67,08,22,68,00,0C
 1082 TP DATA D3,C1,61,1E,08,05,00,02,67,08
 1083 TG DATA 22,68,00,10,D3,C1,61,10,08,05
 1084 I9 DATA 00,03,67,08,22,68,00,14,D3,C1
 1085 9h DATA 61,02,4E,75,70,05,85,91,D3,C3
 1086 uQ DATA 51,C8,FF,FA,4E,75,0C,B9,00,00
 1087 8d DATA 00,01,00,00,3A,92,66,00,00,06
 1088 M9 DATA 60,00,00,6C,04,B9,00,00,00,01
 1089 w2 DATA 00,00,3A,92,41,F9,00,00,3C,3D
 1090 eI DATA 22,39,00,00,3A,92,82,FC,00,64
 1091 hZ DATA 61,00,00,C0,82,FC,00,0A,61,00
 1092 lK DATA 00,B8,61,00,00,B4,60,00,00,3E
 1093 SQ DATA 00,00,3B,7A,3A,3C,00,03,3C,92
 1094 7d DATA 66,00,00,06,60,00,00,2C,06,B9
 1095 iN DATA 00,00,00,01,00,00,3A,92,41,F9
 1096 0e DATA 00,00,3C,3D,22,39,00,00,3A,92
 1097 lF DATA 82,FC,00,64,61,00,00,80,82,FC
 1098 eP DATA 00,0A,61,00,00,78,61,00,00,74
 1099 Lk DATA 4B,F9,00,00,3C,3D,2A,3C,00,00
 1100 lZ DATA 00,03,2C,3C,00,00,00,63,2E,3C
 1101 hO DATA 00,00,00,5D,4E,B9,00,00,04,18
 1102 aI DATA 08,39,00,06,00,BF,E0,01,67,F6
 1103 nZ DATA 60,00,F4,BC,41,F9,00,00,3B,74
 1104 2p DATA 42,81,32,39,00,00,3A,76,82,FC
 1105 ky DATA 00,64,61,00,00,32,82,FC,00,0A
 1106 tB DATA 61,00,00,2A,61,00,00,26,4B,F9
 1107 Nm DATA 00,00,3B,7A,3A,3C,00,03,3C,92
 1108 Qg DATA 01,03,3E,3C,00,C2,23,F9,00,00
 1109 Se DATA 3A,66,00,00,3A,6E,4E,B9,00,00
 1110 UQ DATA 04,18,4E,75,D2,7C,00,30,10,C1
 1111 lT DATA 42,41,48,41,4E,75,42,85,20,79
 1112 a3 DATA 00,00,3B,1C,4E,B9,00,00,2A,CE
 1113 gO DATA E3,5D,20,79,00,00,3B,18,4E,B9
 1114 UF DATA 00,00,2A,CE,E3,5D,20,79,00,00
 1115 9Z DATA 3B,14,4E,B9,00,00,2A,CE,E3,5D
 1116 zJ DATA 20,79,00,00,3B,10,4E,B9,00,00
 1117 Qk DATA 2A,CE,4E,75,26,00,28,01,86,FC
 1118 Q8 DATA 00,08,C8,FC,00,04,D8,43,48,43
 1119 ee DATA 14,3C,00,07,94,03,05,30,40,00
 1120 DZ DATA 67,00,00,06,08,C5,00,00,4E,75
 1121 zh DATA 20,79,00,00,3A,4A,20,39,00,00
 1122 80 DATA 3A,78,90,BC,00,00,00,01,D1,FC
 1123 OT DATA 00,00,02,00,22,3C,00,00,0F,1F
 1124 ON DATA 20,0C,FF,FF,FF,FF,51,C9,FF,FF
 1125 wS DATA 51,C8,FF,EE,4E,75,20,79,00,00
 1126 hV DATA 3A,4A,22,48,24,48,26,48,28,48

1127 Em DATA D3,FC,00,00,00,80,D5,FC,00,00
 1128 4F DATA 01,00,D7,FC,00,00,01,80,D9,FC
 1129 wm DATA 00,00,02,00,20,39,00,00,3A,78
 1130 aK DATA 90,BC,00,00,00,01,22,3C,00,00
 1131 xm DATA 00,1F,42,87,24,18,8E,82,24,19
 1132 pY DATA 8E,82,24,1A,8E,82,24,1B,8E,82
 1133 4b DATA 28,C7,51,C9,FF,EA,D1,FC,00,00
 1134 XP DATA 02,00,D3,FC,00,00,02,00,D5,FC
 1135 zP DATA 00,00,02,00,D7,FC,00,00,02,00
 1136 Lx DATA D9,FC,00,00,02,00,51,C8,FF,C2
 1137 DW DATA 4E,75,4E,B9,00,00,01,10,4E,B9
 1138 Om DATA 00,00,00,06,4E,B9,00,00,00,66
 1139 Wp DATA 4E,B9,00,00,2C,9C,4E,B9,00,00
 1140 52 DATA 01,54,4E,B9,00,00,24,0A,2C,78
 1141 Ha DATA 00,04,22,3C,00,01,00,03,20,39
 1142 to DATA 00,00,3B,20,33,FC,02,00,4E,AE
 1143 sG DATA FF,3A,23,C0,00,00,3A,4A,67,DA
 1144 aV DATA 23,C0,00,00,3B,10,D0,BC,00,00
 1145 Ag DATA 00,80,23,C0,00,00,3B,14,D0,BC
 1146 pC DATA 00,00,00,80,23,C0,00,00,3B,18
 1147 4z DATA D0,BC,00,00,00,80,23,C0,00,00
 1148 dq DATA 3B,1C,D0,BC,00,00,00,80,23,C0
 1149 AP DATA 00,00,3B,20,33,FC,00,01,00,00
 1150 N1 DATA 3A,76,4E,B9,00,00,2A,F2,08,39
 1151 EJ DATA 00,06,00,BF,E0,01,67,00,00,10
 1152 MM DATA 08,39,00,0A,00,DF,0F,16,67,00
 1153 Gu DATA 00,BE,66,E6,22,79,00,00,3A,52
 1154 RP DATA 42,85,42,86,3A,29,00,0E,3C,29
 1155 D5 DATA 00,0C,BA,7C,00,E0,6C,00,DA,86
 1156 9Z DATA BC,7C,00,00,00,60,00,D8,84,61,00
 1157 nq DATA D9,E0,60,BE,2C,78,00,04,20,39
 1158 j7 DATA 00,00,3A,78,C0,FC,02,80,22,79
 1159 ez DATA 00,00,3A,4A,4E,AE,FF,2E,2C,78
 1160 J6 DATA 00,04,20,3C,00,00,08,00,22,79
 1161 x0 DATA 00,00,3A,5E,4E,AE,FF,2E,4E,B9
 1162 G2 DATA 00,00,00,F0,4E,B9,00,00,00,42
 1163 pk DATA 4E,F9,00,00,01,3C,2C,78,00,04
 1164 Nt DATA 20,3C,00,00,08,00,22,3C,00,01
 1165 RK DATA 00,03,4E,AE,FF,3A,23,C0,00,00
 1166 QH DATA 3A,5E,22,40,20,7C,00,00,36,3E
 1167 JN DATA 20,3C,00,00,01,FF,22,D8,51,C8
 1168 5p DATA FF,FC,22,39,00,00,3A,5E,41,F9
 1169 7f DATA 00,00,34,36,D1,FC,00,00,00,0A
 1170 kg DATA 20,3C,00,00,00,00,19,D3,90,D1,FC
 1171 nF DATA 00,00,00,14,51,C8,FF,F6,4E,75
 1172 aL DATA 22,79,00,00,3A,52,3A,29,00,0E
 1173 zM DATA 3C,29,00,0C,BA,7C,00,E0,64,00
 1174 gN DATA 00,2E,BC,7C,00,E0,64,00,00,26
 1175 5t DATA 33,F9,00,00,3A,72,00,00,3A,80
 1176 n0 DATA 33,FC,00,00,00,00,3A,72,61,00
 1177 x5 DATA D9,18,33,F9,00,00,19,D3,90,D1,FC
 1178 LN DATA 3A,72,60,00,FE,EC,BA,7C,01,3F
 1179 hh DATA 66,00,FE,E4,BC,7C,00,00,66,00
 1180 YH DATA FE,DC,60,00,FF,1A,00,00,00,00
 1181 xR DATA 01,40,01,00,00,05,00,01,00,02
 1182 G2 DATA 00,0F,00,00,00,00,00,00,00,00
 1183 5L DATA 00,00,00,00,00,00,00,00,00,00
 1184 hK DATA 00,00,01,40,01,00,00,01,00,00
 1185 7f DATA 00,00,00,01,18,80,00,00,00,00
 1186 80 DATA 00,00,00,00,00,00,00,00,00,00
 1187 Tk DATA 00,00,00,00,00,00,01,40,01,00
 1188 oY DATA 01,40,01,00,00,0F,00,01,00,01
 1189 sB DATA 00,DF,00,DF,02,03,00,00,00,00
 1190 rC DATA 00,01,18,00,00,00,00,00,00,00
 1191 DT DATA 00,00,00,00,00,00,00,00,00,00
 1192 Og DATA 00,00,00,00,00,DF,00,DF,00,DF
 1193 Q1 DATA 00,DF,00,0F,00,58,00,5E,00,90
 1194 f1 DATA 00,2D,02,03,00,00,00,00,00,01
 1195 Jv DATA 18,00,00,00,30,46,00,00,00,00
 1196 IY DATA 00,00,00,00,00,00,00,00,00,00
 1197 Ov DATA 00,00,00,90,00,2D,00,90,00,2D
 1198 yg DATA 00,0F,00,00,2E,76,00,60,00,57
 1199 DL DATA 00,20,00,0A,00,00,08,01,00,04
 1200 4b DATA 00,00,2E,22,00,00,00,00,00,00
 1201 ze DATA 00,00,00,00,00,00,00,00,2E,46
 1202 Ug DATA 00,01,00,00,00,00,00,00,00,00
 1203 BK DATA 03,00,00,05,00,00,2E,32,00,00
 1204 gd DATA 00,00,FF,FE,FF,FE,00,20,FF,FE
 1205 Lx DATA 00,20,00,09,FF,FE,00,09,FF,FE
 1206 CE DATA FF,FE,00,00,2E,6A,00,00,2E,70
 1207 bv DATA 00,00,00,04,00,00,00,00,00,00
 1208 Uk DATA 00,00,00,00,00,00,00,00,00,00
 1209 z6 DATA 00,00,00,00,00,00,00,00,30,30
 1210 Ih DATA 31,00,00,00,30,30,31,00,00,00

1211 Bv DATA 00,00,00,00,00,27,00,75,00,90
 1212 Vx DATA 00,0A,00,00,00,01,00,04,00,00
 1213 Dz DATA 2E,A2,00,00,00,00,00,00,00,00
 1214 Fe DATA 00,00,00,00,00,00,2E,C6,00,02
 1215 h0 DATA 00,00,00,00,00,00,00,00,03,00
 1216 bE DATA 00,05,00,00,2E,B2,00,00,00,00
 1217 hD DATA FF,FE,FF,FE,00,90,FF,FE,00,90
 1218 WE DATA 00,09,FF,FE,00,09,FF,FE,FF,FE
 1219 IX DATA 00,00,2E,EA,00,00,2F,14,00,00
 1220 e0 DATA 00,28,00,00,00,00,00,00,00,00
 1221 hx DATA 00,00,00,00,00,00,00,00,00,00
 1222 Vb DATA 00,00,00,00,00,00,4F,62,6A,65
 1223 Zh DATA 6B,74,65,2E,41,4F,45,00,00,00
 1224 k0 DATA 00,00,00,00,00,00,00,00,00,00
 1225 11 DATA 00,00,00,00,00,00,00,00,00,00
 1226 Qu DATA 00,00,00,00,00,00,00,00,4F,62
 1227 Ug DATA 6A,65,6B,74,65,2E,41,4F,45,00
 1228 o4 DATA 00,00,00,00,00,00,00,00,00,00
 1229 p5 DATA 00,00,00,00,00,00,00,00,00,00
 1230 q6 DATA 00,00,00,00,00,00,00,00,00,00
 1231 6q DATA 00,00,2F,9A,00,38,00,57,00,20
 1232 Tq DATA 00,0A,00,00,08,01,00,04,00,00
 1233 Kr DATA 2E,22,00,00,00,00,00,00,00,00
 1234 fv DATA 00,00,00,00,00,00,2F,6A,00,01
 1235 xI DATA 00,00,00,00,00,00,2F,8E,00,00
 1236 Ny DATA 2F,94,00,00,00,04,00,00,00,00
 1237 xD DATA 00,00,00,00,00,00,00,00,00,00
 1238 yE DATA 00,00,00,00,00,00,00,00,00,00
 1239 Fa DATA 30,30,31,00,00,00,30,30,31,00
 1240 hF DATA 00,00,00,00,00,2F,60,00,88,00,57
 1241 t1 DATA 00,20,00,0A,00,00,08,01,00,04
 1242 kH DATA 00,00,2E,22,00,00,00,00,00,00
 1243 fY DATA 00,00,00,00,00,00,00,00,2F,C6
 1244 5v DATA 00,04,00,00,00,00,00,00,2F,EA
 1245 zH DATA 00,00,2F,F0,00,00,00,04,00,00
 1246 6M DATA 00,00,00,00,00,00,00,00,00,00
 1247 7N DATA 00,00,00,00,00,00,00,00,00,00
 1248 KM DATA 00,00,30,30,31,00,00,00,30,30
 1249 Qm DATA 31,00,00,00,00,00,00,00,00,20
 1250 zk DATA 00,75,00,90,00,0A,00,00,00,01
 1251 21 DATA 00,04,00,00,2E,A2,00,00,00,00
 1252 CS DATA 00,00,00,00,00,00,00,00,00,00
 1253 p7 DATA 30,22,00,05,00,00,00,00,00,00
 1254 Y3 DATA 2E,EA,00,00,2F,14,00,00,00,28
 1255 FV DATA 00,00,00,00,00,00,00,00,00,00
 1256 GW DATA 00,00,00,00,00,00,00,00,00,00
 1257 Ce DATA 00,00,00,00,00,00,00,00,00,39
 1258 mu DATA 00,0F,00,20,00,0A,00,00,08,01
 1259 g7 DATA 00,04,00,00,2E,22,00,00,00,00
 1260 Ka DATA 00,00,00,00,00,00,00,00,00,00
 1261 EX DATA 30,72,00,01,00,00,00,00,00,00
 1262 U7 DATA 30,96,00,00,30,9C,00,00,00,04
 1263 Nd DATA 00,00,00,00,00,00,00,00,00,00
 1264 6W DATA 00,00,00,00,00,00,00,00,00,64
 1265 1R DATA 00,00,00,00,31,30,30,00,00,00
 1266 MH DATA 31,30,30,00,00,00,00,00,30,FE
 1267 Ex DATA 00,38,00,61,00,20,00,0A,00,00
 1268 Uc DATA 08,01,00,04,00,00,2E,22,00,00
 1269 TJ DATA 00,00,00,00,00,00,00,00,00,00
 1270 Cd DATA 00,00,30,CE,00,01,00,00,00,00
 1271 f7 DATA 00,00,30,F2,00,00,30,F8,00,00
 1272 uu DATA 00,04,00,00,00,00,00,00,00,00
 1273 Xn DATA 00,00,00,00,00,00,00,00,00,00
 1274 MT DATA 00,01,00,00,00,00,30,30,31,00
 1275 HS DATA 00,00,30,30,31,00,00,00,00,00
 1276 Ke DATA 00,00,08,00,61,00,20,00,0A,00
 1277 VF DATA 00,00,08,01,00,04,00,00,2E,22
 1278 es DATA 00,00,00,00,00,00,00,00,00,00
 1279 Jk DATA 00,00,00,00,31,2A,00,04,00,00
 1280 2j DATA 00,00,00,00,31,4E,00,00,31,54
 1281 n7 DATA 00,00,00,04,00,00,00,00,00,00
 1282 gw DATA 00,00,00,00,00,00,00,00,00,00
 1283 FO DATA 00,00,00,02,00,00,00,00,30,30
 1284 dz DATA 32,00,00,00,30,30,32,00,00,00
 1285 Bq DATA 00,00,31,B6,00,00,00,57,00,20
 1286 L1 DATA 00,0A,00,00,08,01,00,04,00,00
 1287 CJ DATA 2E,22,00,00,00,00,00,00,00,00
 1288 yS DATA 00,00,00,00,00,00,31,86,00,01
 1289 3q DATA 00,00,00,00,00,00,31,AA,00,00
 1290 bH DATA 31,B0,00,00,00,04,00,00,00,00
 1291 p5 DATA 00,00,00,00,00,00,00,00,00,00
 1292 tA DATA 00,00,00,00,00,01,00,00,00,00
 1293 7S DATA 30,30,31,00,00,00,30,30,31,00
 1294 9z DATA 00,00,00,00,32,12,00,88,00,57

1295 1t DATA 00,20,00,0A,00,00,08,01,00,04
 1296 e9 DATA 00,00,2E,22,00,00,00,00,00,00
 1297 1D DATA 00,00,00,00,00,00,00,00,31,E2
 1298 A6 DATA 00,04,00,00,00,00,00,00,32,06
 1299 6o DATA 00,00,32,0C,00,00,00,04,00,00
 1300 yE DATA 00,00,00,00,00,00,00,00,00,00
 1301 7P DATA 00,00,00,00,00,00,00,02,00,00
 1302 HK DATA 00,00,30,30,32,00,00,00,30,30
 1303 Cx DATA 32,00,00,00,00,00,00,00,7C
 1304 e6 DATA 00,6B,00,20,00,0A,00,00,08,01
 1305 Qr DATA 00,04,00,00,2E,22,00,00,00,00
 1306 4K DATA 00,00,00,00,00,00,00,00,00,00
 1307 gm DATA 32,3E,00,04,00,00,00,00,00,00
 1308 Eu DATA 32,62,00,00,32,68,00,00,00,04
 1309 7N DATA 00,00,00,00,00,00,00,00,00,00
 1310 DU DATA 00,00,00,00,00,00,00,00,01
 1311 m7 DATA 00,00,00,00,30,30,31,00,00,00
 1312 bG DATA 30,30,31,00,00,00,00,00,05,55
 1313 5U DATA 09,99,0B,BB,0E,EE,00,00,00,00
 1314 CS DATA 00,00,00,00,00,00,00,00,00,00
 1315 DT DATA 00,00,00,00,00,00,00,00,00,00
 1316 5C DATA 00,07,00,0F,07,7F,07,07,0F,0F
 1317 Ab DATA 0F,7F,03,33,0A,AA,0F,FF,0F,00
 1318 9p DATA 0F,00,0F,77,00,70,00,0F,07,F7
 1319 K2 DATA 00,02,00,E8,00,0C,00,F8,00,10
 1320 uV DATA 00,E8,00,1A,00,F8,00,1E,00,E8
 1321 pG DATA 00,2B,00,00,00,2C,00,E8,00,36
 1322 IY DATA 00,F8,00,3A,00,E8,00,44,00,F8
 1323 Nj DATA 00,48,00,E8,00,52,00,F8,00,56
 1324 3r DATA 00,E8,00,60,00,F8,00,64,00,E8
 1325 7r DATA 00,6E,00,F8,00,72,00,E8,00,7C
 1326 KG DATA 00,F8,00,80,00,E8,00,8A,00,F8
 1327 k1 DATA 00,8E,00,E8,00,98,00,F8,00,9C
 1328 kK DATA 00,E8,00,A6,00,F8,00,AA,00,E8
 1329 Ym DATA 00,B4,00,F8,00,B8,00,E8,00,C2
 1330 8N DATA 00,F8,00,C6,00,E8,00,D0,00,F8
 1331 1A DATA 00,D4,00,E8,00,DE,00,FF,0F,00
 1332 2L DATA 00,50,01,18,00,5F,01,09,00,2E
 1333 JX DATA 01,18,00,3D,01,1A,00,3F,01,29
 1334 vS DATA 00,4E,00,F8,00,3F,01,07,00,4E
 1335 Xy DATA 01,09,00,3F,01,18,00,4E,01,00
 1336 wd DATA 00,66,01,0F,00,75,01,11,00,66
 1337 Iq DATA 01,20,00,75,01,00,00,77,01,0F
 1338 19 DATA 00,86,01,11,00,77,01,20,00,86
 1339 tS DATA 01,00,00,88,01,0F,00,97,01,11
 1340 KZ DATA 00,88,01,20,00,97,01,00,00,99
 1341 JK DATA 01,0F,00,A8,01,11,00,99,01,20
 1342 d1 DATA 00,A8,00,F8,00,BC,00,0F,C3
 1343 hJ DATA 01,20,00,BC,01,27,00,C3,00,F8
 1344 w2 DATA 00,E2,01,02,00,E9,00,FB,00,FE
 1345 4r DATA 01,02,00,FD,01,0B,00,E2,01,12
 1346 Ef DATA 00,E9,01,0B,00,F6,01,12,00,FD
 1347 5m DATA 01,1B,00,E2,01,22,00,E9,01,1B
 1348 xK DATA 00,F6,01,22,00,FD,01,01,00,BC
 1349 Oq DATA 01,25,00,C3,00,00,00,07,42,00,00
 1350 kS DATA 07,A0,00,00,07,FA,00,00,08,48
 1351 db DATA 00,00,08,96,00,00,0A,28,00,00
 1352 mT DATA 14,6E,00,00,14,A4,00,00,15,38
 1353 tS DATA 00,00,15,84,00,00,16,02,00,00
 1354 x0 DATA 19,A0,00,00,1A,F4,00,00,20,46
 1355 v7 DATA 00,00,20,A2,00,00,21,00,00,00
 1356 bG DATA 21,3C,00,00,21,78,00,00,21,B4
 1357 96 DATA 00,00,21,F0,00,00,22,28,00,00
 1358 f7 DATA 22,60,00,00,00,00,00,10,00,10
 1359 PL DATA 00,02,00,00,00,00,03,00,00,00
 1360 Au DATA 34,4A,00,00,FF,DE,00,10,00,10
 1361 EZ DATA 00,02,00,00,00,40,03,00,00,00
 1362 sM DATA 34,5E,00,11,FF,EF,00,10,00,10
 1363 jn DATA 00,02,00,00,00,00,80,03,00,00
 1364 Kp DATA 34,72,FF,EF,FF,EF,00,10,00,10
 1365 7M DATA 00,02,00,00,00,C0,03,00,00,00
 1366 b9 DATA 34,86,00,00,FF,EF,00,10,00,10
 1367 eZ DATA 00,02,00,00,01,00,03,00,00,00
 1368 BT DATA 00,00,00,00,00,00,00,10,00,10
 1369 hh DATA 00,02,00,00,00,40,03,00,00,00
 1370 1S DATA 34,AE,00,00,FF,DE,00,10,00,10
 1371 bX DATA 00,02,00,00,00,00,03,00,00,00
 1372 Q7 DATA 34,C2,FF,EF,FF,EF,00,10,00,10
 1373 tx DATA 00,02,00,00,00,00,80,03,00,00
 1374 pW DATA 34,D6,00,11,FF,EF,00,10,00,10
 1375 HW DATA 00,02,00,00,00,C0,03,00,00,00
 1376 sv DATA 34,EA,00,00,FF,EF,00,10,00,10

Listing 1. (Fortsetzung)

1377 uv DATA 00,02,00,00,01,40,03,00,00,00
 1378 Ld DATA 00,00,00,00,00,00,00,10,00,10
 1379 49 DATA 00,02,00,00,01,80,03,00,00,00
 1380 7N DATA 35,12,00,11,00,00,00,10,00,10
 1381 S1 DATA 00,02,00,00,01,00,03,00,00,00
 1382 Zf DATA 35,26,00,00,00,11,00,10,00,10
 1383 xv DATA 00,02,00,00,02,00,03,00,00,00
 1384 sE DATA 35,3A,00,11,00,11,00,10,00,10
 1385 79 DATA 00,02,00,00,02,40,03,00,00,00
 1386 KW DATA 35,4E,00,00,00,22,00,10,00,10
 1387 HN DATA 00,02,00,00,02,80,03,00,00,00
 1388 oD DATA 35,62,00,11,00,22,00,10,00,10
 1389 fw DATA 00,02,00,00,02,00,03,00,00,00
 1390 GV DATA 35,76,00,00,00,33,00,10,00,10
 1391 A9 DATA 00,02,00,00,03,00,03,00,00,00
 1392 Z4 DATA 35,8A,00,11,00,33,00,10,00,10
 1393 KN DATA 00,02,00,00,03,40,03,00,00,00
 1394 eA DATA 00,00,00,00,00,00,00,08,00,08
 1395 Ub DATA 00,02,00,00,03,80,03,00,00,00
 1396 7v DATA 35,82,00,00,00,00,00,08,00,08
 1397 o4 DATA 00,02,00,00,03,A0,03,00,00,00
 1398 iE DATA 00,00,00,00,00,00,00,08,00,08
 1399 uc DATA 00,02,00,00,03,C0,03,00,00,00
 1400 ZW DATA 35,DA,00,00,00,14,00,08,00,08
 1401 OK DATA 00,02,00,00,03,E0,03,00,00,00
 1402 wC DATA 35,EE,00,10,00,00,00,08,00,08
 1403 yG DATA 00,02,00,00,03,00,03,00,00,00
 1404 1G DATA 36,02,00,10,00,14,00,08,00,08
 1405 40 DATA 00,02,00,00,03,E0,03,00,00,00
 1406 5F DATA 36,16,00,20,00,00,00,08,00,08
 1407 2K DATA 00,02,00,00,03,C0,03,00,00,00
 1408 U2 DATA 36,2A,00,20,00,14,00,08,00,08
 1409 8S DATA 00,02,00,00,03,E0,03,00,00,00
 1410 TO DATA 00,00,00,00,07,FE,7B,FE,7B,FE
 1411 hx DATA 7B,FE,7B,FE,7B,FE,7B,FE,7B,FE
 1412 EQ DATA 5F,FE,6F,FE,77,FE,7B,FE,7D,FE
 1413 G5 DATA 7E,FE,00,00,00,01,00,21,00,21
 1414 m2 DATA 00,21,00,21,00,21,00,21,00,21
 1415 OX DATA 00,21,00,05,00,09,00,11,00,21
 1416 bt DATA 00,41,00,81,7F,FF,00,00,7F,7E
 1417 FF DATA 7F,FE,7F,FE,7F,FE,7F,FC,7F,FA
 1418 S1 DATA 7F,DE,7F,DE,7F,DE,7F,DE,7F,DE
 1419 Aw DATA 7F,DE,7F,DE,7F,DE,00,00,01
 1420 b0 DATA 01,01,02,01,04,01,08,01,10,01
 1421 Pd DATA 20,01,04,01,04,01,04,01,04,01
 1422 v1 DATA 04,01,04,01,04,01,04,01,7F,FF
 1423 si DATA 00,00,07,FE,7F,FE,7F,FE,7F,FE
 1424 ku DATA 7F,FE,7F,FE,7F,FE,7F,FC,7F,FA
 1425 u0 DATA 00,76,7F,EE,7F,DE,7F,BE,7F,FE
 1426 Yz DATA 00,00,00,01,00,01,00,41,00,21
 1427 Xx DATA 00,11,7F,89,00,05,00,03,00,01
 1428 Mc DATA 00,01,00,01,00,01,00,01,00,01
 1429 Iq DATA 00,01,7F,FF,00,00,7F,FE,7D,FE
 1430 Ia DATA 7B,FE,77,FE,6E,00,5F,FE,3F,FE
 1431 JZ DATA 7F,FE,7F,FE,7F,FE,7F,FE,7F,FE
 1432 u3 DATA 7F,FE,7F,FE,00,00,00,01,00,01
 1433 Rh DATA 00,01,00,01,00,01,00,01,00,01
 1434 dl DATA 00,01,40,01,20,01,11,FF,08,01
 1435 pY DATA 04,01,02,01,00,01,7F,FF,00,00
 1436 MX DATA 7F,BE,7F,BE,7F,DE,7F,BE,6F,7E
 1437 Ls DATA 17,FE,FB,FE,7F,DE,7F,ES,7E,F6
 1438 Dd DATA 7D,FE,7F,FE,7F,FE,7F,FE,00,00
 1439 HN DATA 00,01,02,01,02,01,04,01,02,01
 1440 xG DATA 01,09,00,17,00,21,04,01,68,01
 1441 9S DATA 10,81,00,41,00,21,00,41,00,41
 1442 Dg DATA 7F,FF,00,00,7F,7E,7F,BE,7F,DE
 1443 bb DATA 7F,BE,6F,BE,53,FE,3F,FE,7F,FC
 1444 f3 DATA 7F,CA,7D,F6,7D,FE,7B,FE,7D,FE
 1445 iR DATA 7E,FE,00,00,00,01,01,01,02,01
 1446 5k DATA 04,01,02,01,02,09,00,35,00,03
 1447 dt DATA 40,01,2C,01,10,41,00,41,00,21
 1448 5t DATA 00,41,00,81,7F,FF,00,00,7F,FE
 1449 hD DATA 47,CE,5F,F6,5F,FA,5F,FA,5F,FA
 1450 gw DATA 5F,FA,5F,FA,5F,FA,5F,FA,5F,FA
 1451 LZ DATA 5F,FA,40,02,7F,FE,00,00,00,01
 1452 Jj DATA 00,01,07,F1,07,49,07,45,07,C5
 1453 3J DATA 00,05,00,05,00,05,00,05,00,05
 1454 Gx DATA 00,05,00,05,1F,FD,00,01,7F,FF
 1455 wb DATA 00,00,07,FE,03,86,3F,BE,3B,B6
 1456 7q DATA 3B,B6,3B,B6,3B,B6,3B,BE,3B,AE
 1457 Qo DATA 3B,AE,22,36,7F,F6,7F,FE,7F,FE
 1458 Yc DATA 00,00,00,01,00,01,00,01,3E,39
 1459 To DATA 22,25,22,25,22,25,22,25,22,39
 1460 qv DATA 22,29,22,29,22,25,3F,E5,00,01

1461 kN DATA 00,01,7F,FF,00,00,7E,FE,7E,FE
 1462 Cj DATA 7E,FE,7E,EE,68,96,5E,FA,3E,FC
 1463 v1 DATA 7E,FE,7E,FE,7E,FE,7E,FE,7E,FE
 1464 JQ DATA 7E,FE,7E,FE,00,00,00,01,00,01
 1465 Xn DATA 00,81,00,81,00,81,00,81,00,81
 1466 Gp DATA 00,81,40,83,20,85,16,E9,08,91
 1467 h4 DATA 00,81,00,81,00,81,7F,FF,00,00
 1468 ps DATA 7E,FE,7D,FE,7B,FE,77,FE,7B,FE
 1469 1k DATA 7B,FE,00,02,7F,FE,7B,FE,7B,FE
 1470 Ca DATA 77,FE,7B,FE,7D,FE,7E,FE,00,00
 1471 7y DATA 00,01,00,81,00,41,00,21,00,11
 1472 Lr DATA 00,21,00,21,00,01,3F,FF,00,21
 1473 7Z DATA 00,21,00,11,00,21,00,01,00,81
 1474 zp DATA 7F,FF,00,00,7F,FE,77,FE,6F,FE
 1475 fu DATA 5F,FE,6F,FE,6F,FE,6F,FE,6F,FE
 1476 JI DATA 6F,FE,6F,FE,6F,FE,6F,FE,60,02
 1477 F5 DATA 7F,FE,00,00,00,01,00,01,04,01
 1478 kz DATA 02,01,01,01,02,01,02,01,02,01
 1479 r1 DATA 02,01,02,01,03,FD,00,05,00,05
 1480 VU DATA 00,01,00,11,00,21,00,01,00,81
 1481 Ox DATA 40,0E,5F,EE,5F,FE,5F,EE,5F,EE
 1482 G1 DATA 5F,EE,5F,EE,5F,EE,5F,EE,48,0E
 1483 cu DATA 7F,FE,7F,FE,7F,FE,00,00,00,01
 1484 Gx DATA 00,01,00,01,00,01,0F,FD,08,05
 1485 9P DATA 08,05,08,05,08,05,08,05,08,05
 1486 eY DATA 08,05,08,05,0F,FD,00,01,7F,FF
 1487 UO DATA 00,00,00,01,7F,FF,00,00,7B,FE
 1488 40 DATA 7F,FE,78,3E,7D,FE,7D,FE,7D,FE
 1489 5S DATA 7D,FE,7D,FE,71,DE,7F,FE,7F,FE
 1490 WA DATA 00,00,00,01,00,01,00,01,01,C1
 1491 aS DATA 03,E1,01,C1,00,01,03,C1,01,C1
 1492 Ed DATA 01,C1,01,C1,01,C1,01,C1,07,F1
 1493 g1 DATA 00,01,7F,FF,00,00,7F,FE,63,FE
 1494 FH DATA 5D,FE,5D,FE,5D,FE,63,FE,77,FE
 1495 aS DATA 78,3E,77,FE,47,FE,47,FE,77,FE
 1496 Fu DATA 77,FE,77,FE,00,00,01,00,01,01
 1497 Bg DATA 00,01,00,31,00,49,00,49,00,31
 1498 1C DATA 00,21,00,11,00,09,00,05,00,05
 1499 7U DATA 00,05,07,FD,00,01,7F,FF,00,00
 1500 cS DATA 6E,00,52,00,3E,00,7E,00,7E,00
 1501 FL DATA 7E,00,00,00,01,00,01,00,01,00
 1502 qP DATA 01,00,41,00,2D,00,11,00,7F,00
 1503 yK DATA 00,00,7E,00,7E,00,7E,00,7C,00
 1504 kd DATA 4A,00,76,00,00,00,01,00,09,00
 1505 1Q DATA 35,00,03,00,01,00,01,00,01,00
 1506 1p DATA 7F,00,00,00,76,00,7A,00,7C,00
 1507 yT DATA 7A,00,7A,00,7E,00,00,00,01,00
 1508 OF DATA 11,00,21,00,41,00,21,00,21,00
 1509 uQ DATA 01,00,7F,00,00,00,7E,00,5E,00
 1510 Zn DATA 5E,00,3E,00,5E,00,6E,00,00,00
 1511 Tz DATA 01,00,01,00,05,00,05,00,03,00
 1512 RO DATA 05,00,09,00,7F,00,00,00,00,00
 1513 Pf DATA 00,00,00,00,00,00,00,00,00,00
 1514 Qg DATA 00,00,00,00,00,00,00,00,00,00
 1515 Rh DATA 00,00,00,00,00,00,00,00,00,00
 1516 S1 DATA 00,00,00,00,00,00,00,00,00,00
 1517 Yp DATA 00,00,00,00,00,00,00,00,00,01
 1518 Wn DATA 00,00,00,01,00,00,00,00,00,00
 1519 V1 DATA 00,00,00,00,00,00,00,00,00,00
 1520 Wm DATA 00,00,00,00,00,00,00,00,00,00
 1521 h2 DATA 00,00,00,05,00,00,00,00,00,00
 1522 Yo DATA 00,00,00,00,00,00,00,00,00,00
 1523 Zp DATA 00,00,00,00,00,00,00,00,00,00
 1524 aq DATA 00,00,00,00,00,00,00,00,00,00
 1525 br DATA 00,00,00,00,00,00,00,00,00,00
 1526 es DATA 00,00,00,00,00,00,00,00,00,00
 1527 dt DATA 00,00,00,00,00,00,00,00,00,00
 1528 eu DATA 00,00,00,00,00,00,00,00,00,00
 1529 fv DATA 00,00,00,00,00,00,00,00,00,00
 1530 gw DATA 00,00,00,00,00,00,00,00,00,00
 1531 hx DATA 00,00,00,00,00,00,00,00,00,00
 1532 nm DATA 00,00,00,00,00,00,00,00,00,04
 1533 3Q DATA 00,20,00,05,00,00,00,00,00,00
 1534 k0 DATA 00,00,00,00,00,00,00,00,00,00
 1535 11 DATA 00,00,00,00,00,00,00,00,00,00
 1536 CK DATA 00,00,00,00,00,00,00,00,67,72
 1537 P9 DATA 61,70,68,69,63,73,2E,6C,69,62
 1538 TM DATA 72,61,72,79,00,64,6F,73,2E,6C
 1539 OP DATA 69,62,72,61,72,79,00,69,6E,74
 1540 d0 DATA 75,69,74,69,6F,6E,2E,6C,69,62
 1541 29 DATA 72,61,72,79,00,4F,62,6A,65,6B
 1542 FN DATA 74,20,4E,72,2E,52,20,47,20,42
 1543 Fs DATA 30,20,30,20,30,20,30,31,4C
 1544 E1 DATA 61,64,65,6E,53,70,65,69,63,68

1545 VC DATA 65,72,6E,4C,61,64,65,6E,20,6E
 1546 x8 DATA 61,63,68,20,4F,62,6A,65,6B,74
 1547 AR DATA 44,61,74,65,69,6E,61,6D,65,46
 1548 ZS DATA 65,72,74,69,67,53,70,65,69,63
 1549 00 DATA 68,65,72,6E,20,76,6F,6E,20,4F
 1550 QV DATA 62,6A,65,6B,74,6E,61,63,68,41
 1551 YH DATA 4D,49,47,41,20,4F,42,4A,45,43
 1552 oM DATA 54,45,44,49,54,4F,52,20,56,31
 1553 aN DATA 2E,32,77,72,69,74,74,65,6E,20
 1554 32 DATA 31,39,38,38,20,62,79,4D,49,43
 1555 Fk DATA 48,41,45,4C,20,42,45,52,54,53
 1556 K3 DATA 43,48,49,20,4D,61,72,6B,74,20
 1557 KC DATA 26,20,54,65,63,68,6E,69,6B,5A
 1558 N6 DATA 65,69,67,65,20,4F,62,6A,65,6B
 1559 1U DATA 74,20,4E,72,2E,41,62,62,72,75
 1560 1r DATA 63,68,41,6E,69,6D,61,74,69,6F
 1561 V1 DATA 6E,20,76,6F,6E,20,4F,62,6A,65
 1562 dJ DATA 6B,74,47,65,73,63,68,77,69,6E
 1563 GZ DATA 64,69,67,6B,65,69,74,30,30,35
 1564 bb DATA 4D,61,78,2E,20,4F,62,6A,65,6B
 1565 b0 DATA 74,7A,61,68,6C,4B,6F,70,69,65
 1566 qz DATA 72,65,6E,20,76,6F,6E,20,4F,62
 1567 0a DATA 6A,65,6B,74,41,6E,7A,61,68,6C
 1568 3I DATA 00,00,03,EC,00,00,03,56,00,00
 1569 wX DATA 00,00,00,00,00,02,00,00,00,0C
 1570 51 DATA 00,00,00,18,00,00,00,1E,00,00
 1571 yZ DATA 00,2A,00,00,00,30,00,00,00,3C
 1572 Ne DATA 00,00,00,48,00,00,00,52,00,00
 1573 oI DATA 00,5C,00,00,00,68,00,00,00,72
 1574 CT DATA 00,00,00,7C,00,00,00,82,00,00
 1575 Kw DATA 00,88,00,00,00,8E,00,00,00,98
 1576 7t DATA 00,00,00,4A,00,00,00,AA,00,00
 1577 Ca DATA 00,B2,00,00,00,BA,00,00,00,C6
 1578 Oh DATA 00,00,00,CC,00,00,00,D2,00,00
 1579 PI DATA 00,D8,00,00,00,E6,00,00,00,F2
 1580 62 DATA 00,00,00,F8,00,00,00,01,02,00,00
 1581 V3 DATA 01,36,00,00,01,3E,00,00,01,48
 1582 pC DATA 00,00,01,56,00,00,01,5C,00,00
 1583 EJ DATA 01,66,00,00,01,6A,00,00,01,70
 1584 14 DATA 00,00,01,7E,00,00,01,94,00,00
 1585 1P DATA 01,A2,00,00,01,AE,00,00,01,BE
 1586 Ew DATA 00,00,01,D8,00,00,01,FC,00,00
 1587 1L DATA 02,0A,00,00,02,28,00,00,02,46
 1588 4N DATA 00,00,02,54,00,00,02,6C,00,00
 1589 JO DATA 02,86,00,00,02,94,00,00,02,B6
 1590 GY DATA 00,00,02,D0,00,00,02,D6,00,00
 1591 Ce DATA 02,E4,00,00,02,FE,00,00,03,0A
 1592 Ko DATA 00,00,03,1C,00,00,03,44,00,00
 1593 1Q DATA 03,5E,00,00,03,64,00,00,03,6A
 1594 OD DATA 00,00,03,70,00,00,03,82,00,00
 1595 yY DATA 03,88,00,00,03,9A,00,00,03,A0
 1596 Qp DATA 00,00,03,B2,00,00,03,B8,00,00
 1597 V7 DATA 03,CA,00,00,03,DC,00,00,03,E2
 1598 g4 DATA 00,00,03,F4,00,00,03,FA,00,00
 1599 qP DATA 04,0C,00,00,04,12,00,00,04,1A
 1600 BG DATA 00,00,04,20,00,00,04,36,00,00
 1601 4w DATA 04,42,00,00,04,50,00,00,04,6C
 1602 CU DATA 00,00,04,7A,00,00,04,8A,00,00
 1603 Vo DATA 04,90,00,00,04,9A,00,00,04,B0
 1604 5w DATA 00,00,04,BE,00,00,04,CC,00,00
 1605 mM DATA 04,DE,00,00,05,04,00,00,05,22
 1606 Qk DATA 00,00,05,26,00,00,05,2C,00,00
 1607 p1 DATA 05,32,00,00,05,3E,00,00,05,62
 1608 B7 DATA 00,00,05,68,00,00,05,74,00,00
 1609 y5 DATA 05,C6,00,00,05,D2,00,00,05,E0
 1610 mR DATA 00,00,05,EE,00,00,05,FC,00,00
 1611 5J DATA 06,02,00,00,06,14,00,00,06,18
 1612 71 DATA 00,00,06,1E,00,00,06,3C,00,00
 1613 r5 DATA 06,7A,00,00,06,84,00,00,06,8E
 1614 RR DATA 00,00,06,9C,00,00,06,A2,00,00
 1615 Ov DATA 06,B0,00,00,06,BE,00,00,06,D8
 1616 A3 DATA 00,00,06,FE,00,00,07,22,00,00
 1617 hR DATA 07,36,00,00,07,3A,00,00,07,46
 1618 30 DATA 00,00,07,50,00,00,07,5A,00,00
 1619 kx DATA 07,64,00,00,07,6E,00,00,07,A4
 1620 yY DATA 00,00,07,AE,00,00,07,B8,00,00
 1621 eJ DATA 07,C2,00,00,07,CC,00,00,07,FE
 1622 tw DATA 00,00,08,08,00,00,08,12,00,00
 1623 XT DATA 08,1C,00,00,08,26,00,00,08,4C
 1624 SC DATA 00,00,08,56,00,00,08,60,00,00
 1625 OF DATA 08,6A,00,00,08,74,00,00,08,9A
 1626 Db DATA 00,00,08,A4,00,00,08,AA,00,00
 1627 RA DATA 08,B0,00,00,08,C8,00,00,08,D2
 1628 Vp DATA 00,00,08,DC,00,00,08,E6,00,00

1629 X2 DATA 08,F0,00,00,09,24,00,00,09,2E
1630 cT DATA 00,00,09,38,00,00,09,42,00,00
1631 7N DATA 09,74,00,00,09,7E,00,00,09,88
1632 yd DATA 00,00,09,92,00,00,09,86,00,00
1633 NL DATA 09,C0,00,00,09,CA,00,00,09,D4
1634 Sg DATA 00,00,09,FC,00,00,0A,02,00,00
1635 SM DATA 0A,08,00,00,0A,20,00,00,0A,2A
1636 fE DATA 00,00,0A,30,00,00,0A,3A,00,00
1637 sz DATA 0A,48,00,00,0A,56,00,00,0A,60
1638 Ky DATA 00,00,0A,6E,00,00,0A,88,00,00
1639 Ty DATA 0A,8C,00,00,0A,92,00,00,0A,A4
1640 uh DATA 00,00,0A,AA,00,00,0A,BC,00,00
1641 p9 DATA 0A,C2,00,00,0A,D4,00,00,0A,EE
1642 sS DATA 00,00,0B,08,00,00,0B,22,00,00
1643 1o DATA 0B,3C,00,00,0B,AA,00,00,0B,B0
1644 1x DATA 00,00,0B,BE,00,00,0B,C4,00,00
1645 PR DATA 0B,CE,00,00,0B,E0,00,00,0B,EA
1646 OR DATA 00,00,0C,00,00,00,0C,06,00,00
1647 k8 DATA 0C,0A,00,00,0C,10,00,00,0C,1A
1648 Iu DATA 00,00,0C,24,00,00,0C,2C,00,00
1649 MJ DATA 0C,3A,00,00,0C,44,00,00,0C,52
1650 e8 DATA 00,00,0C,6C,00,00,0C,86,00,00
1651 SU DATA 0C,AD,00,00,0C,BA,00,00,0C,D4
1652 2Q DATA 00,00,0C,EE,00,00,0D,08,00,00
1653 2I DATA 0D,1A,00,00,0D,20,00,00,0D,32
1654 kJ DATA 00,00,0D,38,00,00,0D,4A,00,00
1655 Uf DATA 0D,64,00,00,0D,6A,00,00,0D,82
1656 IO DATA 00,00,0D,AC,00,00,0D,BC,00,00
1657 YJ DATA 0D,CA,00,00,0D,D0,00,00,0D,D6
1658 Fr DATA 00,00,0D,E4,00,00,0D,EA,00,00
1659 7E DATA 0D,FO,00,00,0D,F6,00,00,0E,04
1660 Ps DATA 00,00,0E,10,00,00,0E,1A,00,00
1661 e2 DATA 0E,48,00,00,0E,62,00,00,0E,6E
1662 mt DATA 00,00,0E,78,00,00,0E,84,00,00
1663 1b DATA 0E,94,00,00,0E,A4,00,00,0E,B6
1664 Ds DATA 00,00,0E,C4,00,00,0E,CA,00,00
1665 Tt DATA 0E,DO,00,00,0E,D6,00,00,0E,E4
1666 uQ DATA 00,00,0E,EA,00,00,0E,FO,00,00
1667 Jv DATA 0E,F6,00,00,0F,04,00,00,0F,0A
1668 pJ DATA 00,00,0F,14,00,00,0F,1A,00,00
1669 On DATA 0F,28,00,00,0F,2E,00,00,0F,48
1670 ri DATA 00,00,0F,6C,00,00,10,0C,00,00
1671 Pv DATA 10,3E,00,00,10,4E,00,00,10,84
1672 7N DATA 00,00,10,8A,00,00,10,94,00,00
1673 QE DATA 10,A6,00,00,10,B0,00,00,10,C6
1674 YF DATA 00,00,10,CC,00,00,10,D0,00,00
1675 ov DATA 10,D6,00,00,10,E0,00,00,10,EA
1676 SI DATA 00,00,10,F2,00,00,11,00,00,00
1677 pE DATA 11,0A,00,00,11,18,00,00,11,32
1678 2W DATA 00,00,11,4C,00,00,11,66,00,00
1679 Gr DATA 11,80,00,00,11,9A,00,00,11,B4
1680 RG DATA 00,00,11,CE,00,00,11,E8,00,00
1681 Eq DATA 11,FA,00,00,12,00,00,00,12,12
1682 8h DATA 00,00,12,18,00,00,12,2A,00,00
1683 zq DATA 12,30,00,00,12,42,00,00,12,5C
1684 hv DATA 00,00,12,62,00,00,12,7A,00,00
1685 ce DATA 12,A4,00,00,12,B4,00,00,12,C2
1686 1l DATA 00,00,12,C8,00,00,12,CE,00,00
1687 xD DATA 12,DC,00,00,12,E2,00,00,12,E8
1688 Ub DATA 00,00,12,EE,00,00,13,00,00,00
1689 zh DATA 13,0E,00,00,13,18,00,00,13,26
1690 xM DATA 00,00,13,2C,00,00,13,32,00,00
1691 B1 DATA 13,3E,00,00,13,44,00,00,13,4A
1692 Ju DATA 00,00,13,50,00,00,13,5A,00,00
1693 Eq DATA 13,66,00,00,13,70,00,00,13,9A
1694 Sn DATA 00,00,13,A0,00,00,13,B0,00,00
1695 Wp DATA 13,CA,00,00,13,D0,00,00,13,E4
1696 1P DATA 00,00,13,EA,00,00,13,FA,00,00
1697 3g DATA 14,00,00,00,14,06,00,00,14,16
1698 g1 DATA 00,00,14,24,00,00,14,2A,00,00
1699 Ho DATA 14,38,00,00,14,3E,00,00,14,48
1700 JI DATA 00,00,14,4E,00,00,14,5C,00,00
1701 7p DATA 14,62,00,00,14,70,00,00,14,76
1702 rz DATA 00,00,14,86,00,00,14,9C,00,00
1703 NX DATA 14,A6,00,00,14,B0,00,00,14,BA
1704 Z1 DATA 00,00,14,C4,00,00,15,3A,00,00
1705 KD DATA 15,44,00,00,15,4E,00,00,15,58
1706 rd DATA 00,00,15,86,00,00,15,90,00,00
1707 Or DATA 15,9A,00,00,15,A4,00,00,16,04
1708 26 DATA 00,00,16,16,00,00,16,20,00,00
1709 pQ DATA 16,36,00,00,16,3C,00,00,16,40
1710 VQ DATA 00,00,16,46,00,00,16,50,00,00
1711 YI DATA 16,5A,00,00,16,62,00,00,16,70
1712 qv DATA 00,00,16,7A,00,00,16,88,00,00

1713 dJ DATA 16,A2,00,00,16,BC,00,00,16,D6
1714 4r DATA 00,00,16,F0,00,00,17,0A,00,00
1715 eY DATA 17,24,00,00,17,3E,00,00,17,58
1716 PM DATA 00,00,17,6A,00,00,17,70,00,00
1717 Sh DATA 17,82,00,00,17,88,00,00,17,9A
1718 en DATA 00,00,17,A0,00,00,17,B2,00,00
1719 tV DATA 17,CC,00,00,17,D2,00,00,17,EA
1720 UQ DATA 00,00,18,14,00,00,18,20,00,00
1721 qX DATA 18,26,00,00,18,34,00,00,18,3A
1722 Ay DATA 00,00,18,40,00,00,18,46,00,00
1723 OB DATA 18,54,00,00,18,62,00,00,18,68
1724 fd DATA 00,00,18,76,00,00,18,7C,00,00
1725 w0 DATA 18,86,00,00,18,8C,00,00,18,98
1726 JY DATA 00,00,18,A2,00,00,18,A8,00,00
1727 uZ DATA 18,B4,00,00,18,C4,00,00,18,CE
1728 Hu DATA 00,00,18,DE,00,00,18,EE,00,00
1729 qh DATA 18,F6,00,00,19,06,00,00,19,18
1730 k0 DATA 00,00,19,1E,00,00,19,26,00,00
1731 CF DATA 19,36,00,00,19,3C,00,00,19,4C
1732 TQ DATA 00,00,19,54,00,00,19,5A,00,00
1733 01 DATA 19,6A,00,00,19,84,00,00,19,8E
1734 V9 DATA 00,00,19,94,00,00,19,A2,00,00
1735 8J DATA 19,A8,00,00,19,B8,00,00,19,C2
1736 ak DATA 00,00,19,D0,00,00,19,DA,00,00
1737 He DATA 19,E8,00,00,19,FE,00,00,1A,02
1738 JC DATA 00,00,1A,08,00,00,1A,1A,00,00
1739 BW DATA 1A,20,00,00,1A,32,00,00,1A,38
1740 7g DATA 00,00,1A,4A,00,00,1A,50,00,00
1741 6J DATA 1A,62,00,00,1A,68,00,00,1A,7A
1742 JR DATA 00,00,1A,80,00,00,1A,92,00,00
1743 HL DATA 1A,AC,00,00,1A,BC,00,00,1A,E2
1744 kR DATA 00,00,1A,EB,00,00,1A,F6,00,00
1745 Ib DATA 1A,FC,00,00,1B,00,00,00,1B,06
1746 R5 DATA 00,00,1B,10,00,00,1B,1A,00,00
1747 If DATA 1B,22,00,00,1B,30,00,00,1B,3A
1748 Dr DATA 00,00,1B,48,00,00,1B,5A,00,00
1749 8f DATA 1B,64,00,00,1B,7A,00,00,1B,94
1750 17 DATA 00,00,1B,AE,00,00,1B,C8,00,00
1751 au DATA 1B,E2,00,00,1B,FC,00,00,1C,0E
1752 Af DATA 00,00,1C,14,00,00,1C,26,00,00
1753 cA DATA 1C,2C,00,00,1C,3E,00,00,1C,58
1754 4e DATA 00,00,1C,68,00,00,1C,8E,00,00
1755 U7 DATA 1C,94,00,00,1C,A2,00,00,1C,B0
1756 XS DATA 00,00,1C,B6,00,00,1C,BC,00,00
1757 ln DATA 1C,C6,00,00,1C,CA,00,00,1C,D4
1758 Bh DATA 00,00,1C,E2,00,00,1C,EB,00,00
1759 Pg DATA 1C,EE,00,00,1C,F8,00,00,1C,FC
1760 Ko DATA 00,00,1D,02,00,00,1D,08,00,00
1761 HY DATA 1D,14,00,00,1D,1A,00,00,1D,20
1762 Ir DATA 00,00,1D,24,00,00,1D,2A,00,00
1763 zQ DATA 1D,34,00,00,1D,3A,00,00,1D,44
1764 V5 DATA 00,00,1D,4E,00,00,1D,56,00,00
1765 AO DATA 1D,60,00,00,1D,6A,00,00,1D,74
1766 yN DATA 00,00,1D,7A,00,00,1D,88,00,00
1767 S5 DATA 1D,96,00,00,1D,B0,00,00,1D,B6
1768 Ve DATA 00,00,1D,BC,00,00,1D,CE,00,00
1769 o9 DATA 1D,D4,00,00,1D,E6,00,00,1D,EC
1770 4J DATA 00,00,1D,FE,00,00,1E,04,00,00
1771 T1 DATA 1E,16,00,00,1E,30,00,00,1E,3A
1772 5J DATA 00,00,1E,44,00,00,1E,72,00,00
1773 8E DATA 1E,8C,00,00,1E,A6,00,00,1E,AC
1774 zV DATA 00,00,1E,B6,00,00,1E,C0,00,00
1775 D1 DATA 1E,CA,00,00,1E,D4,00,00,1E,DE
1776 Kd DATA 00,00,1E,E2,00,00,1E,F0,00,00
1777 T4 DATA 1E,F6,00,00,1F,00,00,00,1F,1E
1778 sL DATA 00,00,1F,24,00,00,1F,2A,00,00
1779 kM DATA 1F,56,00,00,1F,66,00,00,1F,E6
1780 sn DATA 00,00,1F,F2,00,00,1F,FC,00,00
1781 AH DATA 20,06,00,00,20,10,00,00,20,1A
1782 g0 DATA 00,00,20,20,00,00,20,26,00,00
1783 ZJ DATA 20,30,00,00,20,34,00,00,20,3E
1784 Ht DATA 00,00,20,48,00,00,20,5E,00,00
1785 k5 DATA 20,68,00,00,20,72,00,00,20,7C
1786 m9 DATA 00,00,20,86,00,00,20,8E,00,00
1787 rh DATA 20,94,00,00,20,9A,00,00,20,A4
1788 Ou DATA 00,00,20,AA,00,00,20,BC,00,00
1789 fh DATA 20,C6,00,00,20,D0,00,00,20,DA
1790 XD DATA 20,F2,00,00,20,E4,00,00,20,EC,00,00
1791 hg DATA 20,F2,00,00,20,F8,00,00,21,04
1792 sS DATA 00,00,21,10,00,00,21,2E,00,00
1793 1U DATA 21,34,00,00,21,40,00,00,21,4C
1794 p6 DATA 00,00,21,6A,00,00,21,70,00,00
1795 8t DATA 21,7C,00,00,21,88,00,00,21,A6
1796 2w DATA 00,00,21,AC,00,00,21,B8,00,00

1797 EM DATA 21,C4,00,00,21,E2,00,00,21,E8
1798 md DATA 00,00,21,F4,00,00,22,00,00,00
1799 B1 DATA 22,1A,00,00,22,20,00,00,22,2C
1800 Xo DATA 00,00,22,38,00,00,22,52,00,00
1801 GT DATA 22,58,00,00,22,62,00,00,22,68
1802 xB DATA 00,00,22,72,00,00,22,7C,00,00
1803 K7 DATA 22,84,00,00,22,92,00,00,22,9C
1804 vS DATA 00,00,22,B2,00,00,22,C8,00,00
1805 yv DATA 22,D6,00,00,22,FA,00,00,23,14
1806 Cm DATA 00,00,23,2C,00,00,23,46,00,00
1807 4p DATA 23,4C,00,00,23,5E,00,00,23,64
1808 DB DATA 00,00,23,76,00,00,23,90,00,00
1809 r1 DATA 23,96,00,00,23,AE,00,00,23,D8
1810 D1 DATA 00,00,23,E4,00,00,23,EA,00,00
1811 ZB DATA 23,F8,00,00,24,06,00,00,24,0C
1812 Zz DATA 00,00,24,12,00,00,24,1C,00,00
1813 SL DATA 24,20,00,00,24,26,00,00,24,2C
1814 OJ DATA 00,00,24,3C,00,00,24,42,00,00
1815 WA DATA 24,4E,00,00,24,5A,00,00,24,66
1816 nk DATA 00,00,24,72,00,00,24,78,00,00
1817 8e DATA 24,7E,00,00,24,84,00,00,24,92
1818 Zc DATA 00,00,24,98,00,00,24,A6,00,00
1819 q8 DATA 24,B4,00,00,24,BA,00,00,24,C0
1820 Ga DATA 00,00,24,C6,00,00,24,D0,00,00
1821 rC DATA 24,DA,00,00,24,E2,00,00,24,FO
1822 73 DATA 00,00,24,FA,00,00,25,10,00,00
1823 b0 DATA 25,26,00,00,25,34,00,00,25,58
1824 im DATA 00,00,25,72,00,00,25,8A,00,00
1825 S3 DATA 25,A4,00,00,25,AA,00,00,25,BC
1826 ev DATA 00,00,25,C2,00,00,25,D4,00,00
1827 r1 DATA 25,EE,00,00,25,F4,00,00,26,0C
1828 WY DATA 00,00,26,36,00,00,26,42,00,00
1829 7U DATA 26,48,00,00,26,56,00,00,26,64
1830 Ws DATA 00,00,26,6A,00,00,26,6E,00,00
1831 PG DATA 26,74,00,00,26,7A,00,00,26,80
1832 XI DATA 00,00,26,92,00,00,26,A0,00,00
1833 OY DATA 26,A6,00,00,26,C6,00,00,26,D4
1834 nY DATA 00,00,26,DE,00,00,26,EE,00,00
1835 lg DATA 26,F4,00,00,27,00,00,00,27,0C
1836 1r DATA 00,00,27,18,00,00,27,24,00,00
1837 1a DATA 27,2C,00,00,27,32,00,00,27,36
1838 GS DATA 00,00,27,3C,00,00,27,42,00,00
1839 WO DATA 27,4C,00,00,27,50,00,00,27,5E
1840 tF DATA 00,00,27,6E,00,00,27,7C,00,00
1841 Vy DATA 27,9C,00,00,27,A8,00,00,27,AE
1842 Kd DATA 00,00,27,B4,00,00,27,D6,00,00
1843 oC DATA 27,DC,00,00,27,EC,00,00,28,06
1844 hx DATA 00,00,28,2A,00,00,28,38,00,00
1845 4K DATA 28,56,00,00,28,8C,00,00,28,AA
1846 8Z DATA 00,00,28,CC,00,00,28,D6,00,00
1847 PM DATA 28,E0,00,00,28,EA,00,00,28,F4
1848 pf DATA 00,00,29,50,00,00,29,A0,00,00
1849 8y DATA 29,B2,00,00,29,B8,00,00,29,BE
1850 mh DATA 00,00,29,E0,00,00,29,F2,00,00
1851 wN DATA 29,F8,00,00,29,FE,00,00,2A,18
1852 c3 DATA 00,00,2A,30,00,00,2A,44,00,00
1853 13 DATA 2A,4C,00,00,2A,66,00,00,2A,78
1854 1E DATA 00,00,2A,7C,00,00,2A,82,00,00
1855 Rw DATA 2A,98,00,00,2A,9E,00,00,2A,A6
1856 KO DATA 00,00,2A,AC,00,00,2A,B4,00,00
1857 1k DATA 2A,BA,00,00,2A,C2,00,00,2A,C8
1858 OA DATA 00,00,2A,F4,00,00,2A,FA,00,00
1859 De DATA 2B,22,00,00,2B,48,00,00,2B,96
1860 b5 DATA 00,00,2B,9C,00,00,2B,A2,00,00
1861 VI DATA 2B,A8,00,00,2B,AE,00,00,2B,B4
1862 Gu DATA 00,00,2B,C4,00,00,2B,D2,00,00
1863 9G DATA 2B,DA,00,00,2B,E6,00,00,2B,F2
1864 Eo DATA 00,00,2B,FE,00,00,2C,0A,00,00
1865 40 DATA 2C,12,00,00,2C,18,00,00,2C,38
1866 mI DATA 00,00,2C,64,00,00,2C,6E,00,00
1867 K3 DATA 2C,82,00,00,2C,8C,00,00,2C,92
1868 ar DATA 00,00,2C,98,00,00,2C,B2,00,00
1869 gx DATA 2C,BA,00,00,2C,CC,00,00,2C,D2
1870 pn DATA 00,00,2C,F2,00,00,2D,10,00,00
1871 pC DATA 2D,14,00,00,2D,1C,00,00,2D,26
1872 ur DATA 00,00,2D,2A,00,00,2D,D8,00,00
1873 Op DATA 2D,F6,00,00,2E,08,00,00,2E,18
1874 Co DATA 00,00,2E,2A,00,00,2E,46,00,00
1875 rZ DATA 2E,4A,00,00,2E,88,00,00,2E,98
1876 PI DATA 00,00,2E,AA,00,00,2E,C6,00,00
1877 mh DATA 2E,CA,00,00,2F,3E,00,00,2F,50
1878 4J DATA 00,00,2F,60,00,00,2F,6A,00,00

Listing 1. (Fortsetzung)

```

1879 VV DATA 2F,6E,00,00,2F,9A,00,00,2F,AC
1880 nb DATA 00,00,2F,BC,00,00,2F,C6,00,00
1881 ED DATA 2F,CA,00,00,30,08,00,00,30,18
1882 Us DATA 00,00,30,22,00,00,30,26,00,00
1883 KQ DATA 30,58,00,00,30,68,00,00,30,72
1884 bu DATA 00,00,30,76,00,00,30,A2,00,00
1885 YL DATA 30,B4,00,00,30,C4,00,00,30,CE
1886 x0 DATA 00,00,30,D2,00,00,31,10,00,00
1887 3y DATA 31,20,00,00,31,2A,00,00,31,2E
1888 YD DATA 00,00,31,5A,00,00,31,6C,00,00
1889 3m DATA 31,7C,00,00,31,86,00,00,31,8A
1890 P5 DATA 00,00,31,B6,00,00,31,C8,00,00
1891 F4 DATA 31,D8,00,00,31,E2,00,00,31,E6
1892 xG DATA 00,00,32,24,00,00,32,34,00,00
1893 xg DATA 32,3E,00,00,32,42,00,00,33,DE
1894 vF DATA 00,00,33,E2,00,00,33,E6,00,00
1895 Fr DATA 33,EA,00,00,33,EE,00,00,33,F2

```

```

1896 wS DATA 00,00,33,F6,00,00,33,FA,00,00
1897 Ad DATA 33,FE,00,00,34,02,00,00,34,06
1898 YK DATA 00,00,34,0A,00,00,34,0E,00,00
1899 tX DATA 34,12,00,00,34,16,00,00,34,1A
1900 gB DATA 00,00,34,1E,00,00,34,22,00,00
1901 L1 DATA 34,26,00,00,34,2A,00,00,34,2E
1902 eo DATA 00,00,34,32,00,00,34,46,00,00
1903 rn DATA 34,5A,00,00,34,6E,00,00,34,82
1904 A8 DATA 00,00,34,AA,00,00,34,BE,00,00
1905 Sn DATA 34,D2,00,00,34,E6,00,00,35,0E
1906 jt DATA 00,00,35,22,00,00,35,36,00,00
1907 uq DATA 35,4A,00,00,35,5E,00,00,35,72
1908 7Q DATA 00,00,35,86,00,00,35,AE,00,00
1909 zd DATA 35,D6,00,00,35,EA,00,00,35,FE
1910 oy DATA 00,00,36,12,00,00,36,26,00,00
1911 Eu DATA 00,1D,00,00,00,01,00,00,0E,2C
1912 hE DATA 00,00,0E,3E,00,00,0E,54,00,00

```

```

1913 Gm DATA 0E,68,00,00,0E,AC,00,00,0F,40
1914 ag DATA 00,00,0F,54,00,00,0F,64,00,00
1915 Ea DATA 0F,78,00,00,0F,82,00,00,0F,A0
1916 Qu DATA 00,00,0F,B8,00,00,0F,D0,00,00
1917 D7 DATA 13,7E,00,00,13,86,00,00,13,AA
1918 wH DATA 00,00,13,B6,00,00,15,B6,00,00
1919 bQ DATA 15,EE,00,00,1E,B0,00,00,1E,BA
1920 jU DATA 00,00,1E,C4,00,00,1E,CE,00,00
1921 p4 DATA 1E,D8,00,00,1F,EE,00,00,1F,F8
1922 YD DATA 00,00,20,02,00,00,20,0C,00,00
1923 Qm DATA 20,16,00,00,00,00,00,00,03,F2
1924 w7 DATA 00,00,03,EB,00,00,01,00,00,00
1925 qX DATA 03,F2
(C) 1987 M&T

```

Listing 1. (Schluß)

Fortsetzung von Seite 14

Heiße Schlachten im Amiga

Programmname: Hauptprogramm

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: Amiga-Basic 1.2

Bemerkung: Bitte zuerst das Ladeprogramm (Listing 2) starten, damit genügend Speicher bereitsteht

Programm : hauptprogramm

```

1 hT0 SCREEN 1,640,255,3,2
2 IX WINDOW 2,"Amiga gegen den Rest der Welt",(0,0)-(615,230),16,1
3 Va REM Weltkarte
4 M1 f=1:z=0:start=0:FOR p=1 TO 6:ka(p)=0:NEXT
5 qp za=3:abbruch=0:ab=0:dugmax=7
6 Nw GOTO spielabfr
7 hk namen:
8 yG2 PALETTE 3,.93,.2,0:REM rot
9 iP PALETTE 4,1,1,.1:REM gelb
10 yW PALETTE 5,1,.6,.3:REM orange
11 Xb PALETTE 6,.33,.87,0:REM GRÜN
12 vz PALETTE 7,1,.7,.9:REM rosa
13 kw REM weiß=1
14 IS LOCATE 23,19:PRINT "Abbruch"
15 UN LINE(210,175)-(280,185),1,B:LINE(210,175)-(280,185),1
16 yd LINE(210,185)-(280,175),1
17 Ck PAINT(212,180),3,1:PAINT(258,180),3,1
18 5g PAINT(235,176),5,1:PAINT(235,184),5,1
19 9L LOCATE 21,20:PRINT "Spieler"
20 p8 LINE(210,157)-(280,170),1,B
21 9S COLOR 2,0
22 LP LOCATE 3,2:PRINT "Alaska":LOCATE 2,10:PRINT "NW-"
23 Jc LOCATE 3,10:PRINT "Territorium"
24 u1 LOCATE 2,26:PRINT "Grönland"
25 N5 LOCATE 6,6:PRINT "Alberta"
26 TS LOCATE 6,20:PRINT "Ontario"
27 Jg LOCATE 5,26:PRINT "Quebeck"
28 Xd LOCATE 8,2:PRINT "Weststaaten"
29 Nu LOCATE 9,20:PRINT "Oststaaten"
30 41 LOCATE 11,2:PRINT "Mittel-":LOCATE 12,4:PRINT "amerika"
31 Ea LOCATE 14,2:PRINT "Venezuela"
32 46 LOCATE 17,5:PRINT "Peru"
33 iu LOCATE 17,14:PRINT "Brasilien"
34 NM LOCATE 22,4:PRINT "Argentinien"
35 JU LOCATE 6,31:PRINT "Island"
36 1j LOCATE 5,37:PRINT "Skandinavien"
37 qi LOCATE 10,31:PRINT "GB"
38 8x LOCATE 7,47:PRINT "Ukraine"
39 AL LOCATE 6,55:PRINT "Ural"
40 7v LOCATE 3,60:PRINT "Sibirien"
41 pU LOCATE 2,68:PRINT "Jakutien"
42 Ta LOCATE 6,67:PRINT "Irkutsk"
43 HI LOCATE 10,68:PRINT "Mongolei"
44 Ep LOCATE 7,73:PRINT "Kamt-":LOCATE 8,71:PRINT "schatka"
45 Oj LOCATE 13,60:PRINT "China"
46 U6 LOCATE 15,58:PRINT "Indien"

```

```

47 eW LOCATE 13,46:PRINT "Mittlerer":LOCATE 14,47:PRINT "Osten"
48 J2 LOCATE 11,54:PRINT "Afghanistan"
49 NB LOCATE 15,66:PRINT "Siam"
50 EU LOCATE 17,72:PRINT "Neu":LOCATE 18,72:PRINT "Guinea"
51 GV LOCATE 19,59:PRINT "Indonesien"
52 sY LOCATE 22,59:PRINT "West-":LOCATE 23,60:PRINT "Australien"
53 1s LOCATE 22,73:PRINT "Ost-":LOCATE 25,68:PRINT "Australien"
54 7P LOCATE 11,72:PRINT "Japan"
55 1Z LOCATE 15,28:PRINT "Nord-West-":LOCATE 17,31:PRINT "Afrika"
56 s5 LOCATE 15,40:PRINT "Ägypten"
57 Oz LOCATE 17,45:PRINT "Ostafrika"
58 eG LOCATE 19,42:PRINT "Kongo"
59 6m LOCATE 23,37:PRINT "Südafrika"
60 OY LOCATE 24,52:PRINT "Madagaskar"
61 Ix RETURN
62 tS REM alaska
63 cM0 al:
64 pe4 LOCATE 4,5:PRINT land(L)
65 6o LINE(30,4)-(32,14):LINE-(48,9):LINE-(64,13)
66 4h LINE-(72,16):LINE-(72,40):LINE-(48,35)
67 ux LINE-(40,36)
68 H1 LINE-(24,33):LINE-(16,25):LINE-(24,22)
69 XA LINE-(16,15):LINE-(48,11)
70 91 LINE-(30,4)
71 8N PAINT (33,15),f,2
72 w8 RETURN
73 bt2 REM Nordwest-Territorium
74 qb0 a2:
75 9I4 LOCATE 4,15:PRINT land(L)
76 3s LINE(72,16)-(128,14):LINE-(176,28)
77 BW LINE-(176,35):LINE-(96,35)
78 GE LINE-(72,40)
79 Vn PAINT (75,30),f,2
80 eG RETURN
81 SQ2 REM Grönland
82 In0 a3:
83 JG4 LOCATE 3,32:PRINT land(L)
84 PN LINE(176,28)-(210,15):LINE-(230,20)
85 vT LINE-(200,35):LINE-(176,35)
86 dv LINE(248,14)-(250,5):LINE-(270,6)
87 Nq LINE-(288,20):LINE-(260,40)
88 dT LINE-(248,25):LINE-(248,14)
89 4U PAINT (205,30),f,2
90 jN PAINT(249,15),f,2
91 pR RETURN
92 vH3 REM Alberta
93 F20 a4:
94 Y14 LOCATE 6,14:PRINT land(L)
95 50 LINE(72,40)-(72,55):LINE-(160,55)
96 KE LINE-(160,35)
97 yf PAINT(159,36),f,2
98 wY RETURN
99 S63 REM Ontario
100 PDO a5:
101 lq5 LOCATE 7,22:PRINT land(L)
102 vm LINE(200,35)-(210,60):LINE-(180,55)
103 VR LINE-(160,55)
104 T3 PAINT(161,54),f,2
105 3f RETURN
106 Z53 REM Quebec
107 Z00 a6:
108 2C5 LOCATE 7,27:PRINT land(L)

```



```

109 04      LINE(202,50)-(225,40):LINE-(245,53)
110 AG      LINE-(225,58):LINE-(210,60):LINE-(200,35)
111 10      PAINT(224,57),f,2
112 Am      RETURN
113 1Y3      REM Weststaaten
114 j20 a7:
115 5I5      LOCATE 9,14:PRINT land(L)
116 QS      LINE(72,55)-(70,65):LINE-(73,72)
117 nx      LINE-(80,75):LINE-(130,80)
118 j9      LINE-(145,60):LINE-(170,55)
119 CM      PAINT(72,60),f,2
120 Iu      RETURN
121 sw3      REM Oststaaten
122 u10 a8:
123 Ec5      LOCATE 10,19:PRINT land(L)
124 30      LINE(130,80)-(140,85):LINE-(170,80)
125 Kf      LINE-(172,90):LINE-(180,88)
126 Xt      LINE-(179,80):LINE-(215,70)
127 pl      LINE-(220,58)
128 zZ      PAINT(218,60),f,2
129 R3      RETURN
130 d23      REM Mittelamerika
131 6y0 a9:
132 4L5      LOCATE 11,11:PRINT land(L)
133 MH      LINE(73,72)-(74,75):LINE-(85,90)
134 h1      LINE-(90,98):LINE-(112,105)
135 kB      LINE-(116,104):LINE-(105,95)
136 gn      LINE-(110,91):LINE-(120,87)
137 1y      LINE-(140,85)
138 qP      PAINT(130,84),f,2
139 bD      RETURN
140 pJ3      REM Venezuela
141 KK0 a10:
142 bx5      LOCATE 15,12:PRINT land(L)
143 sH      LINE(116,104)-(128,103):LINE-(180,123)
144 xm      LINE-(148,124):LINE-(118,119)
145 Nr      LINE-(94,124):LINE-(66,119)
146 Ww      LINE-(76,111):LINE-(100,105)
147 BG      LINE-(112,105)
148 4C      PAINT(90,122),f,2
149 lN      RETURN
150 qg3      REM Preu
151 YZ0 a11:
152 zN5      LOCATE 19,10:PRINT land(L)
153 vE      LINE(66,119)-(56,132):LINE-(64,146)
154 AU      LINE-(76,154):LINE-(78,159)
155 Ds      LINE-(114,164):LINE-(122,144)
156 g5      LINE-(88,433):LINE-(94,124)
157 Wy      PAINT(112,162),f,2
158 uW      RETURN
159 Hk3      REM Brasilien
160 lN0 a12:
161 W25      LOCATE 19,18:PRINT land(L)
162 Nv      LINE(122,144)-(136,152):LINE-(134,165)
163 80      LINE-(140,169):LINE-(164,156)
164 6v      LINE-(186,154):LINE-(216,142)
165 jt      LINE-(180,123)
166 tP      PAINT(140,165),f,2
167 3f      RETURN
168 7E3      REM Argentinien
169 y10 a13:
170 uE5      LOCATE 23,11:PRINT land(L)
171 yW      LINE(78,159)-(80,178):LINE-(74,194)
172 z0      LINE-(76,208):LINE-(84,215)
173 h1      LINE-(92,219):LINE-(90,213)
174 rJ      LINE-(92,208):LINE-(98,201)
175 DT      LINE-(140,169)
176 6H      PAINT(88,213),f,2
177 Dp      RETURN
178 ZX3      REM Island
179 CG0 a14:
180 AC5      LOCATE 7,33:PRINT land(L)
181 eB      LINE(255,44)-(273,48):LINE-(283,42)
182 AJ      LINE-(303,51):LINE-(279,58)
183 Hp      LINE-(261,59):LINE-(248,50)
184 76      LINE-(255,44)
185 tW      PAINT(260,57),f,2
186 My      RETURN
187 h73      REM Skandinavien
188 PU0 a15:
189 Z15      LOCATE 8,39:PRINT land(L)
190 ar      LINE(292,72)-(290,64):LINE-(306,57)
191 jB      LINE-(314,48):LINE-(312,41)

```

```

192 ze      LINE-(326,38):LINE-(340,35)
193 Mr      LINE-(350,47):LINE-(354,67)
194 Eq      LINE-(346,61):LINE-(342,48)
195 1J      LINE-(336,54):LINE-(338,66)
196 tA      LINE-(330,74):LINE-(320,76)
197 sM      LINE-(308,69):LINE-(292,72)
198 oN      PAINT(320,74),f,2
199 ZB      RETURN
200 Um3      REM Mitteleuropa
201 gm0 a16:
202 b15      LOCATE 11,38:PRINT land(L)
203 oP      LINE(288,87)-(298,83):LINE-(306,75)
204 Yp      LINE-(312,76):LINE-(318,81)
205 tH      LINE-(346,75):LINE-(360,85)
206 uK      LINE-(312,91):LINE-(288,87)
207 vT      PAINT(312,90),f,2
208 iK      RETURN
209 KH3      REM Westeuropa
210 t00 a17:
211 1A5      LOCATE 13,35:PRINT land(L)
212 Rb      LINE(288,87)-(284,92):LINE-(276,95)
213 Mo      LINE-(280,97):LINE-(264,99)
214 aX      LINE-(268,107):LINE-(286,108)
215 ZA      LINE-(288,101):LINE-(312,98)
216 C7      LINE-(312,91)
217 tA      PAINT(284,107),f,2:PAINT(288,89),f,2
218 sU      RETURN
219 rZ3      REM Südeuropa
220 7F0 a18:
221 j55      LOCATE 12,42:PRINT land(L)
222 Zn      LINE(312,98)-(322,103):LINE-(314,106)
223 Lz      LINE-(332,107):LINE-(334,103)
224 Pm      LINE-(328,96):LINE-(340,97)
225 xc      LINE-(352,102):LINE-(358,99)
226 Yy      LINE-(360,95):LINE-(368,95)
227 Kw      LINE-(360,85):LINE-(360,85)-(312,91)
228 h9      PAINT(330,106),f,2
229 3f      RETURN
230 Ow3      REM Ukraine
231 MV0 a19:
232 QW5      LOCATE 9,49:PRINT land(L)
233 dR      LINE(368,95)-(370,98):LINE-(414,109)
234 Sz      LINE-(412,106):LINE-(412,99)
235 EZ      LINE-(416,97):LINE-(422,81)
236 2U      LINE-(434,80):LINE-(446,62)
237 Tv      LINE-(426,52):LINE-(420,42)
238 JG      LINE-(438,29):LINE-(428,16)
239 ud      LINE-(380,28):LINE-(380,38)
240 L6      LINE-(366,37):LINE-(356,33)
241 9W      LINE-(340,35):LINE-(346,75)-(354,67)
242 bC      PAINT(396,103),f,2
243 Ht      RETURN
244 xP3      REM Groß Britanien
245 340 a20:
246 Mm5      LOCATE 12,28:PRINT land(L)
247 7x      LINE(256,75)-(262,75):LINE-(268,81)
248 De      LINE-(278,88):LINE-(272,92)
249 hz      LINE-(262,93):LINE-(258,90)
250 Qt      LINE-(260,85):LINE-(254,78)
251 QU      LINE-(256,75)
252 Oy      LINE(240,81)-(248,82):LINE-(254,87)
253 Yo      LINE-(252,91):LINE-(246,94)
254 oE      LINE-(234,89):LINE-(236,84)
255 mg      LINE-(240,81)
256 d7      PAINT(263,92),f,2:PAINT(246,93),f,2
257 V7      RETURN
258 Ib3      REM Ural
259 lN0 a21:
260 qp5      LOCATE 8,57:PRINT land(L)
261 Of      LINE(428,16)-(458,13):LINE-(468,4)
262 Vm      LINE-(474,3):LINE-(478,5)
263 HA      LINE-(480,22):LINE-(486,18)
264 Fq      LINE-(492,24):LINE-(496,78)
265 It      LINE-(486,79):LINE-(480,72)
266 2c      LINE-(456,66):LINE-(446,62)
267 C4      PAINT(488,78),f,2
268 gI      RETURN
269 Yt3      REM Afghanistan
270 ad0 a22:
271 lC5      LOCATE 12,56:PRINT land(L)
272 E9      LINE(486,79)-(470,95):LINE-(462,107)

```

Listing 1. (Fortsetzung)

```

273 sg      LINE-(448,113):LINE-(416,107)
274 nt      LINE-(416,97)
275 sP      PAINT(448,110),f,2
276 oQ      RETURN
277 513     REM Mittlerer Osten
278 mq0 a23:
279 F15      LOCATE 15,49:PRINT land(L)
280 MC      LINE(448,113)-(448,118):LINE-(436,118)
281 JU      LINE-(416,113):LINE-(424,119)
282 su      LINE-(398,128):LINE-(394,124)
283 2r      LINE-(374,115):LINE-(374,105)
284 M0      LINE-(356,103):LINE-(362,100)
285 sy      LINE-(370,98)
286 Wx      PAINT(400,125),f,2
287 zb      RETURN
288 p43     REM Sibirien
289 160 a24:
290 6u5      LOCATE 5,63:PRINT land(L)
291 LV      LINE(492,24)-(486,18):LINE-(484,9)
292 me      LINE-(500,8):LINE-(510,3)
293 8B      LINE-(546,1):LINE-(530,29)
294 Jt      LINE-(534,47):LINE-(522,78):LINE-(506,80)
295 dq      LINE-(496,78)
296 Mv      PAINT(500,74),f,2
297 91      RETURN
298 E13     REM Jakutien
299 F10 a25:
300 KC5      LOCATE 3,69:PRINT land(L)
301 hG      LINE(546,1)-(560,6):LINE-(580,3):LINE-(584,23)
302 m6      LINE-(578,30):LINE-(564,38):LINE-(560,28)
303 pn      LINE-(530,29)
304 sT      PAINT(564,30),f,2
305 Ht      RETURN
306 Cn3     REM Irkutsk
307 RY0 a26:
308 gb5      LOCATE 7,68:PRINT land(L)
309 xf      LINE(522,78)-(534,76):LINE-(554,71):LINE-(550,63)
310 TJ      LINE-(564,54):LINE-(564,38)
311 oQ      PAINT(530,65),f,2
312 00      RETURN
313 A13     REM Kamtschatka
314 ck0 a27:
315 bL5      LOCATE 5,73:PRINT land(L)
316 sI      LINE(580,3)-(594,6):LINE-(606,13):LINE-(602,18)
317 mS      LINE-(612,26):LINE-(610,36):LINE-(594,51)
318 tm      LINE-(598,66):LINE-(592,69):LINE-(584,62)
319 J5      LINE-(584,54):LINE-(574,65):LINE-(566,71)
320 A1      LINE-(570,79):LINE-(562,86):LINE-(554,71)
321 Ds      PAINT(552,64),f,2
322 YA      RETURN
323 Gp3     REM Mongolei
324 qz0 a28:
325 d55      LOCATE 11,67:PRINT land(L)
326 H1      LINE(562,86)-(568,92):LINE-(552,91):LINE-(526,87)
327 lw      LINE-(506,80):LINE-(562,86)-(554,71)
328 wV      PAINT(512,80),f,2
329 fH      RETURN
330 9f3     REM China
331 1B0 a29:
332 oG5      LOCATE 13,65:PRINT land(L)
333 K7      LINE(462,107)-(482,107):LINE-(510,119):LINE-(528,114)
334 qH      LINE-(552,116):LINE-(548,122):LINE-(560,119)
335 PQ      LINE-(566,108):LINE-(560,98):LINE-(552,91)
336 6w      PAINT(494,79),f,2
337 nP      RETURN
338 MS3     REM Indien
339 ce0 a30:
340 wM5      LOCATE 16,60:PRINT land(L)
341 I3      LINE(448,118)-(456,120):LINE-(470,132):LINE-(480,140)
342 p1      LINE-(490,131):LINE-(510,125):LINE-(510,119)
343 Cp      PAINT(480,138),f,2
344 uW      RETURN
345 Ha3     REM Siam
346 nq0 a31:
347 In5      LOCATE 16,65:PRINT land(L)
348 V2      LINE(510,125)-(524,130):LINE-(530,133):LINE-(530,129)
349 JV      LINE-(546,141):LINE-(544,136):LINE-(550,137)
350 Sy      LINE-(560,134):LINE-(558,136):LINE-(548,122)
351 9J      PAINT(542,136),f,2
352 2e      RETURN
353 Wa3     REM Japan
354 z30 a32:
355 Ac5      LOCATE 13,74:PRINT land(L)
356 zx      LINE(582,81)-(588,88):LINE-(586,94):LINE-(592,103)

```

```

357 Sp      LINE-(582,108):LINE-(572,104):LINE-(574,98)
358 Bq      LINE-(576,92):LINE-(574,84):LINE-(582,81)
359 En      PAINT(580,106),f,2
360 Am      RETURN
361 7V3     REM Neu Guinea
362 BG0 a33:
363 mK5      LOCATE 19,74:PRINT land(L)
364 87      LINE(574,137)-(584,137):LINE-(602,139):LINE-(610,144)
365 M6      LINE-(614,159):LINE-(594,157):LINE-(580,148)
366 Jg      LINE-(574,141):LINE-(574,137)
367 Gp      PAINT(610,157),f,2
368 Iu      RETURN
369 3r3     REM Indonesien
370 NTO a34:
371 7J5      LOCATE 19,69:PRINT land(L)
372 nw      LINE(564,141)-(572,146):LINE-(576,152):LINE-(564,159)
373 1e      LINE-(538,157):LINE-(532,153):LINE-(534,149)
374 C0      LINE-(540,145):LINE-(564,141)
375 2u      LINE(514,150)-(524,152):LINE-(534,159):LINE-(528,161)
376 yI      LINE-(518,160):LINE-(514,154):LINE-(514,150)
377 Wr      PAINT(564,158),f,2:PAINT(530,159),f,2
378 S46     RETURN
379 Et3     REM Westaustralien
380 b10 a35:
381 JE5      LOCATE 22,68:PRINT land(L)
382 mu      LINE(572,163)-(558,162):LINE-(542,167):LINE-(532,168)
383 XG      LINE-(516,179):LINE-(522,188):LINE-(536,193)
384 8z      LINE-(552,190):LINE-(568,185):LINE-(578,177)
385 14      LINE-(572,163)
386 6m      PAINT(538,191),f,2
387 bD      RETURN
388 OT3     REM Ostaustralien
389 ow0 a36:
390 nG5      LOCATE 23,74:PRINT land(L)
391 w5      LINE(572,163)-(582,159):LINE-(606,166):LINE-(614,174)
392 wJ      LINE-(610,184):LINE-(598,191):LINE-(568,185)
393 tW      PAINT(600,189),f,2
394 iK      RETURN
395 ad3     REM Nord-West-Afrika
396 z80 a37:
397 Ae5      LOCATE 18,35:PRINT land(L)
398 7J      LINE(308,112)-(304,109):LINE-(272,110):LINE-(258,117)
399 mx      LINE-(250,130):LINE-(260,144):LINE-(284,150)
400 OS      LINE-(300,149):LINE-(308,157):LINE-(318,156)
401 Vp      LINE-(326,147):LINE-(340,144):LINE-(328,140)
402 Bd      LINE-(332,135):LINE-(298,125):LINE-(296,121)
403 gq      LINE-(308,112)
404 qN      PAINT(262,143),f,2
405 tV      RETURN
406 4u3     REM Ägypten
407 E00 a38:
408 4U5      LOCATE 16,42:PRINT land(L)
409 5f      LINE(308,112)-(334,114):LINE-(352,113):LINE-(374,115)
410 XG      LINE-(380,124):LINE-(332,135)
411 1F      PAINT(332,133),f,2
412 Oc      RETURN
413 DK3     REM Ostafrika
414 Pa0 a39:
415 Uz5      LOCATE 18,45:PRINT land(L)
416 2w      LINE(380,124)-(384,127):LINE-(396,130):LINE-(406,129)
417 VG      LINE-(394,145):LINE-(398,157):LINE-(386,170)
418 1R      LINE-(352,163):LINE-(356,155):LINE-(350,145)
419 p0      LINE-(340,144)
420 tc      PAINT(386,166),f,2
421 91      RETURN
422 Yr3     REM Kongo
423 140 a40:
424 rB5      LOCATE 20,41:PRINT land(L)
425 N6      LINE(352,163)-(332,163):LINE-(320,160):LINE-(308,159)
426 Zs      LINE-(308,157)
427 8r      LINE(300,157)-(318,156):LINE-(326,147)
428 5b      PAINT(350,162),f,2
429 Ht      RETURN
430 Ma3     REM Südafrika
431 DH0 a41:
432 6R5      LOCATE 22,40:PRINT land(L)
433 72      LINE(308,159)-(314,173):LINE-(312,182):LINE-(326,195)
434 Hf      LINE-(342,206):LINE-(354,202):LINE-(368,191)
435 12      LINE-(386,170)
436 Bf      PAINT(342,204),f,2
437 P1      RETURN
438 RJ3     REM Madagaskar
439 PU0 a42:
440 Om5      LOCATE 23,51:PRINT land(L)

```



```

441 IO LINE(410,171)-(414,176):LINE-(412,185):LINE-(400,193)
442 1K LINE-(392,193):LINE-(390,187):LINE-(402,173)
443 1A LINE-(410,171)
444 09 PAINT(398,192),f,2
445 X9 RETURN
446 zr3 REM *** Spieleranzahl ***
447 KE1 spielabfr:
448 Tp5 INPUT "Wieviele menschliche Spieler (0-6)";sp
449 d07 IF sp=6 THEN GOTO w1
450 II IF sp>6 THEN PRINT "zu viel !!!":GOTO spielabfr
451 c05 PRINT "Wieviele Spieler soll der Computer übernehmen (m
ax.;6-sp;")";:INPUT " ",csp
452 6v7 IF csp+sp>6 THEN csp=6-sp
453 c13 w1:
454 N16 spges=sp+csp;z=1
455 nX FOR I=1 TO spges
456 529 IF I>sp THEN Spieler$(I)=CHR$(70+z):z=z+1
457 UZ IF I<sp+1 THEN Spieler$(I)=CHR$(64+I)
458 TY6 NEXT
459 aH2 DIM land(43),land$(43)
460 nv FOR I=1 TO 42:land(I)=1:land$(I)="":NEXT
461 c15 RANDOMIZE TIMER
462 QU2 FOR D=1 TO spges
463 eF3 landanz=INT(42/spges)
464 MU FOR dd=1 TO landanz
465 qF rndw:
466 wI4 x=INT(RND*44):IF x>42 THEN x=42
467 aU IF land$(x)<>"*" THEN GOTO rndw
468 v2 land$(x)=Spieler$(D)
469 Fj3 NEXT:NEXT
470 zB5 z=1
471 vc3 FOR I=1 TO 42
472 Cv IF land$(I)="*" THEN land$(I)=Spieler$(z):z=z+1
473 in NEXT
474 3B2 w3:
475 38 IF sp=0 THEN GOTO w4
476 wk4 PRINT :PRINT "Bitte nacheinander die Namen eingeben"
477 80 PRINT " Höchstens 8 Buchstaben bitte !"
478 bd2 FOR D=1 TO sp
479 Ps4 LOCATE 5+D,5:LINE INPUT N$(D)
480 pu NEXT
481 3y2 LOCATE 14,2
482 tk PRINT "Farbtabelle:"
483 D94 LOCATE 16,1
484 Da PRINT "weiß=1 rot=3 gelb=4 orange=5 grün=6 rosa=7"
485 r05 LINE(0,118)-(390,128),,B:LINE(55,118)-(55,128)
486 16 LINE(110,118)-(110,128):LINE(175,118)-(175,128)
487 SX LINE(255,118)-(255,128):LINE(320,118)-(320,128)
488 mG4 LOCATE 18,5:PRINT "Bitte eine Nummer wählen !"
489 mo2 FOR D=1 TO sp
490 la3 LOCATE 19+D,1
491 7J PRINT N$(D);:INPUT " welche Farbe";farb(D)
492 Wk5 IF farb(D)=1 THEN LOCATE 16,1:PRINT " "
493 BJ IF farb(D)=3 THEN LOCATE 16,9:PRINT " "
494 mD IF farb(D)=4 THEN LOCATE 16,16:PRINT " "
495 ZK IF farb(D)=5 THEN LOCATE 16,24:PRINT " "
496 sV IF farb(D)=6 THEN LOCATE 16,34:PRINT " "
497 pc IF farb(D)=7 THEN LOCATE 16,42:PRINT " "
498 7C3 NEXT
499 SJ IF sp=6 OR csp=0 THEN GOTO w5
500 JJ REM *Computer-Namen*
501 Xg2 w4:
502 hr5 FOR D=sp+1 TO spges
503 hS READ cname$
504 QQ N$(D)=cname$
505 EJ NEXT
506 Gn DATA "Amiga","Atari","Napoleon","Hirni"
507 OV DATA "007","Fred"
508 tU3 REM *Computer-Farben*
509 6c5 IF sp=0 THEN GOTO far2
510 79 FOR D=1 TO sp
511 R2 farb2(D)=farb(D)
512 LQ NEXT
513 ep y=1
514 GD RESTORE fdaten
515 tq3 falesen:
516 QN5 READ fa:z=0
517 K1 IF fa=20 THEN GOTO w6
518 FH FOR D=1 TO sp
519 R0 IF fa<>farb2(D) THEN z=z+1
520 Q3 IF z=sp THEN farb3(y)=fa:y=y+1
521 Xm NEXT D:z=0:GOTO falesen

```

```

522 OF4 fdaten:
523 Mw7 DATA 1,3,4,5,6,7,20
524 7E4 far2:
525 RO6 RESTORE fdaten
526 Iy FOR p=1 TO spges:READ fa
527 7J farb(p)=fa:NEXT
528 oL GOTO nbl
529 5G0 w6:
530 v65 y=1
531 AK FOR D=sp+1 TO spges
532 Wx farb(D)=farb3(y):y=y+1
533 g1 NEXT
534 7H0 w5:
535 vL1 REM Nachbarländer einlesen
536 MK nbl:
537 8H DIM Nb(42,8),Lname$(42)
538 103 RESTORE Nachbarland
539 Jo FOR L=1 TO 42
540 qn5 FOR p=1 TO 8
541 eu READ N
542 F1 Nb(L,p)=N
543 q7 NEXT p,L
544 Uq2 RESTORE mausposition
545 Pu4 FOR L=1 TO 42
546 sS6 READ A,B,c,c,L$
547 bM Lname$(L)=L$
548 v0 NEXT
549 pQ2 REM *** Reihenfolge ***
550 f85 FOR p=1 TO spges:rfs(p)="+":NEXT
551 7L2 FOR p=1 TO spges
552 cd rfg:
553 6F RANDOMIZE TIMER
554 11 x=INT(RND(1)*spges+1)
555 qQ IF x=0 THEN GOTO rfg
556 w0 IF x>spges THEN x=spges
557 6G4 IF rfs(x)<>"+" THEN GOTO rfg
558 15 rfs(x)=Spieler$(p)
559 6B NEXT
560 RV2 WINDOW 4,,(250,130)-(400,135),0,1
561 Yq WINDOW OUTPUT 2
562 ue CLS:GOSUB namen:beg=0
563 hC FOR L=1 TO 42
564 Cf4 IF land$(L)=Spieler$(1) THEN f=farb(1)
565 Jo IF land$(L)=Spieler$(2) THEN f=farb(2)
566 Qx IF land$(L)=Spieler$(3) THEN f=farb(3)
567 X6 IF land$(L)=Spieler$(4) THEN f=farb(4)
568 eF IF land$(L)=Spieler$(5) THEN f=farb(5)
569 10 IF land$(L)=Spieler$(6) THEN f=farb(6)
570 Am GOSUB v6
571 NW3 NEXT:GOTO v7
572 ku0 v6:
573 k23 WINDOW OUTPUT 2
574 34 ON L GOSUB a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a1
4,a15,a16,a17,a18,a19,a20,a21,a22,a23,a24,a25,a26,a27,a2
8,a29,a30,a31,a32,a33,a34,a35,a36,a37,a38,a39,a40,a41,a4
2
575 Lf5 GOSUB f4
576 Vr IF beg=0 AND L=1 THEN CLS:PRINT " Nordamerika";
577 Ua IF beg=0 AND L=9 THEN CLS:PRINT " Südamerika";:GOSUB p
ause
578 QJ IF beg=0 AND L=13 THEN CLS:PRINT " Europa";:GOSUB paus
e
579 4C IF beg=0 AND L=20 THEN CLS:PRINT " A s i e n";:GOSUB p
ause
580 5q IF beg=0 AND L=32 THEN CLS:PRINT " Australien";:GOSUB
pause
581 rr IF beg=0 AND L=36 THEN CLS:PRINT " A f r i k a";:beg=1
582 km3 RETURN
583 942 END
584 zA0 v7:
585 sn1 REM **** D A S S P I E L B E G I N N T ****
586 jQ4 WINDOW CLOSE 3
587 072 MENU 1,0,1,"Aktionen"
588 7w MENU 1,1,1,"Angriff"
589 9c MENU 1,2,1,"Verschieben"
590 18 MENU 1,3,1,"Tauschen"
591 EV MENU 1,4,1,"nächster Spieler"
592 GR MENU 1,5,1,"ende"
593 17 DIM leer$(80),vs(42)
594 6A1 start:

```

Listing 1. (Fortsetzung)

```

595 eW3 WINDOW 4,,(0,200)-(620,235),0,1
596 JT WINDOW OUTPUT 4
597 sE r=1:dug=1
598 u3 beginn:
599 oX5 Karte=0:vt=1:leer$=""
      "
600 k4 GOSUB f4
601 G9 GOSUB welcherSpieler
602 fA GOSUB wievielArmeen
603 la3 IF rf$(r)>"F" THEN GOTO computer
604 lC spiel:
605 rd5 ON MENU GOSUB Spieler:MENU ON
606 dd GOTO spiel
607 oA3 Spieler:
608 JL5 v=MENU(1):CLS
609 je ON v GOSUB angriff,verschieben,tauschen,nextSpieler,en
      de
610 uE GOSUB f4
611 Dp RETURN
612 K4 angriff:
613 i17 GOSUB f4:PRINT "Von"
614 SY5 angr1:
615 F17 GOSUB welchesLand
616 G29 IF abbruch=1 THEN GOSUB f4:CLS:abbruch=0:RETURN
617 i67 IF land$(L)<>rf$(r) THEN
618 NC9 CLS:LOCATE 3,5
619 RE PRINT"Dies ist nicht Dein Land ! Noch mal.":GOTO a
      ngriff
620 bK ELSE
621 xq END IF
622 IX7 IF vt=1 THEN RETURN
623 dt9 IF land(L)<2 THEN
624 QEB CLS:LOCATE 2,5
625 Wk PRINT"Hier gibt es keine Angriffsarmeen !"
626 GD GOSUB pause:CLS:GOTO angriff
627 iR ELSE
628 4x END IF
629 6d7 AusgLand$=la$:Anr=L
630 it LOCATE 1,1
631 eM PRINT "Von ";AusgLand$;" nach "
632 j15 ziel:
633 XJ7 GOSUB welchesLand
634 YK9 IF abbruch=1 THEN GOSUB f4:CLS:abbruch=0:RETURN
635 hk7 IF land$(L)=rf$(r) THEN
636 KH9 LOCATE 3,5
637 QN PRINT"Dies ist Dein eigenes Land ! Noch mal.":GOT
      O ziel
638 tc ELSE
639 F8 END IF
640 SPB FOR p=1 TO 8
641 XaD IF L=Nb(Anr,p) THEN GOTO z2
642 RWB NEXT
643 RO LOCATE 3,5
644 D4 PRINT "Dies ist kein Nachbarland ! Noch mal.":G
      OTO ziel
645 r16 z2:
646 2L7 Zielland$=la$:Znr=L:
647 e2 LOCATE 3,5:PRINT "
      "
648 OB LOCATE 1,1
649 K3 PRINT "Von ";AusgLand$;" nach ";Zielland$
650 iW FOR p=1 TO spges
651 R19 IF Spieler$(p)=land$(Znr) THEN
652 u7D Znamen$=N$(p):fz=farb(p)
653 8r ELSE
654 UN END IF
655 eJ9 NEXT
656 8K7 z4:
657 Pc FOR I=1 TO 42:vs(I)=0:NEXT
658 9M z=0:GOSUB wuerfel:GOSUB auswertung
659 os9 IF ka spi=6 THEN
660 biB CLS:PRINT "Du hast 6 Karten und mußt tauschen !"
661 J2 GOSUB pause:GOSUB wKarte
662 HO ELSE
663 dW END IF
664 8U9 IF land(Anr)<2 OR land$(Znr)=rf$(r) THEN RETURN
665 Ve7 LOCATE 3,2
666 o2 PRINT "Soll der Angriff wiederholt werden ?";
667 4S LOCATE 5,37:PRINT "Ja Nein";
668 XJ LINE(280,30)-(310,40),,B:LINE(330,30)-(370,40),,B
669 IT z3:
670 239 m=MOUSE(0):IF MOUSE(0)=0 THEN z3
671 os IF MOUSE(3)>280 AND MOUSE(3)<315 THEN

```

```

672 zZB CLS:GOSUB pause
673 ph LOCATE 1,1:PRINT "Von ";AusgLand$;" nach ";Ziell
      and$
674 QB GOTO z4
675 UD ELSE
676 qJ END IF
677 iR CLS:RETURN
678 gY5 welchesLand:
679 WD7 m=MOUSE(0):IF MOUSE(0)=0 THEN GOTO welchesLand
680 4p hori=MOUSE(3):verti=MOUSE(4)
681 Df LOCATE 3,1:PRINT leer$;
682 i4 RESTORE mausposition
683 d8 FOR L=1 TO 42
684 ON READ h,h2,v,v2,la$
685 NK IF hori>210 AND hori<280 AND verti>175 AND verti<
      185 THEN abbruch=1:RETURN
686 qH IF hori>h AND hori<h2 AND verti>v AND verti<v2 T
      HEN RETURN
687 AF NEXT
688 EC LOCATE 4,5
689 Mp PRINT "Noch mal ";:m=MOUSE(0)
690 8u6 wLm:
691 ST8 IF MOUSE(0)=0 THEN GOTO wLm
692 G1 hori=MOUSE(3):verti=MOUSE(4)
693 2r LOCATE 4,5:PRINT " ";
694 iF LOCATE 3,1:PRINT " "
695 No GOTO welchesLand
696 8F5 welcherSpieler:
697 Th7 FOR p=1 TO spges
698 hq IF Spieler$(p)=rf$(r) THEN namen$=N$(p):f=farb(p):sp
      i=p
699 MR NEXT
700 Pd9 g=0:FOR p=1 TO spges
701 9K IF rf$(p)="*" THEN g=g+1
702 PU NEXT
703 Yn IF g=spges-1 THEN GOTO sieg
704 OO7 WINDOW OUTPUT 2:LOCATE 21,28:PRINT " "
705 HA COLOR f,0:LOCATE 21,28:PRINT namen$:COLOR 2,0
706 Pq GOSUB f4:CLS:B=500:GOSUB Soldat
707 pI COLOR 1,0:PRINT namen$;" ist am Zug.";
708 mO RETURN
709 iW5 wievielArmeen:
710 267 armee=0:S=0:la=0:FOR p=1 TO 42
711 i4 IF land$(p)=rf$(r) THEN la=la+1:S=S+1
712 MI9 IF p=9 THEN
713 6iB IF S=9 THEN armee=armee+5
714 b9 S=0
715 8r ELSE
716 UN END IF
717 wo9 IF p=13 THEN
718 JSB IF S=4 THEN armee=armee+2
719 gE S=0
720 Dw ELSE
721 ZS END IF
722 uk9 IF p=20 THEN
723 CmB IF S=7 THEN armee=armee+5
724 iJ S=0
725 iI ELSE
726 eX END IF
727 709 IF p=32 THEN
728 OLB IF S=12 THEN armee=armee+7
729 qO S=0
730 N6 ELSE
731 jc END IF
732 OL9 IF p=36 THEN
733 yhB IF S=4 THEN armee=armee+2
734 vT S=0
735 SB ELSE
736 oh END IF
737 JD9 IF p=42 THEN
738 xgB IF la=0 THEN rf$(r)="*":GOTO ende
739 Ev IF S=6 THEN armee=armee+3
740 iZ S=0
741 YH ELSE
742 un END IF
743 de7 NEXT:armeen=INT(la/3)
744 u8 IF armeen<3 THEN armeen=3
745 iW armeen=armeen+armee:LOCATE 2,1
746 Ka PRINT namen$;" erhält ";armeen;" Armeen."
747 bc9 IF rf$(r)>"F" THEN RETURN
748 LM7 vt=1:LOCATE 3,1
749 HQ PRINT "Zeige mit der Maus auf eines Deiner Länder !"
750 nI vert:

```



```

751 tC9      GOSUB angr1
752 B3      WINDOW 4,,(0,200)-(620,235),0,1
753 gS      GOSUB f4:Ainp=0
754 CD7      eing:
755 Ju9      LOCATE 1,1
756 6F      INPUT "Wieviel Armeen in dieses Land";arminp
757 IH      Ainp=Ainp+arminp
758 tK      IF Ainp>armeen THEN
759 NcC      PRINT "Mehr als Du bekommst !"
760 XO      Ainp=Ainp-arminp
761 Wh      FOR p=1 TO 3000:NEXT:CLS
762 UA      GOTO eing
763 ud      ELSE
764 G9      END IF
765 q8A      WINDOW OUTPUT 2
766 O9      land(L)=land(L)+arminp:GOSUB v6
767 op      GOSUB f4:CLS
768 rg      IF Ainp=armeen THEN vt=0:RETURN
769 NG      PRINT namen$;" Du hast noch ";armeen-Ainp;" Armee
n."
770 Oh      armeen=armeen-Ainp
771 cE      PRINT "Auf welches Land !"
772 JT      GOTO vert
773 oG5      wuerfel:
774 W17      LINE(400,5)-(615,42),1,B:PAINT(402,6),2,1
775 Jv      LINE(510,5)-(510,42),1:COLOR 1,2
776 hq      RANDOMIZE TIMER
777 Dy      LOCATE 2,55:PRINT namen$:LOCATE 2,67:PRINT Znamen$
778 t3      w=INT(RND*7):RANDOMIZE TIMER
779 Xn      w2=INT(RND*7):RANDOMIZE TIMER
780 RB      w3=INT(RND*7)
781 VV9      IF w=0 OR w2=0 OR w3=0 THEN GOTO wuerfel
782 gw      IF w>6 OR w2>6 OR w3>6 THEN GOTO wuerfel
783 FI      IF w>=w2 AND w>=w3 THEN wu1(z)=w:GOTO wf1
784 9MA      IF w>=w2 AND w<=w3 THEN wu2(z)=w:GOTO wf2
785 AN      IF w>=w3 AND w<=w2 THEN wu2(z)=w:GOTO wf2
786 4R      IF w<=w2 AND w<=w3 THEN wu3(z)=w:GOTO wf3
787 5JB      wf1:
788 rrD      IF w2>=w3 THEN
789 mWF      wu2(z)=w2:wu3(z)=w3
790 L4      ELSE
791 oYH      wu3(z)=w2:wu2(z)=w3
792 lbF      END IF
793 KXB      GOTO wurf
794 GV      wf2:
795 IpD      IF w>=w2 THEN
796 pYF      wu3(z)=w2:wu1(z)=w3
797 SB      ELSE
798 raH      wu3(z)=w3:wu1(z)=w2
799 pIF      END IF
800 ReD      GOTO wurf
801 RhB      wf3:
802 lzD      IF w2<=w3 THEN
803 saF      wu2(z)=w2:wu1(z)=w3
804 ZI      ELSE
805 ucH      wu2(z)=w3:wu1(z)=w2
806 wpF      END IF
807 Ow6      wurf:
808 Oy7      IF z=1 THEN GOTO zwurf
809 Sw9      IF land(Anr)<3 THEN Wanz=1:GOTO wurf1
810 dA      IF land(Anr)<4 THEN Wanz=2:GOTO wurf2
811 hO      Wanz=3:GOTO wurfw
812 vN7      zwurf:
813 TJ9      IF land(Znr)<2 THEN GOTO wurf1
814 fS      IF land(Znr)<3 THEN GOTO wurf2
815 6n7      GOTO wurfw
816 V3      wurf1:
817 ey9      wu1(z)=wu2(z)
818 Mc      wu2(z)=0
819 e87      wurf2:
820 S39      wu3(z)=0
821 MO7      wurfw:
822 Zd9      IF z=1 THEN GOTO wurfw2
823 PEB      LOCATE 3,53:PRINT "1.";
824 aM      LOCATE 4,53:PRINT "2.";
825 lU      LOCATE 5,53:PRINT "3.";
826 I39      LOCATE 3,57:PRINT wu1(0):SOUND 200,.5
827 HA      IF wu2(0)>0 THEN LOCATE 4,57:PRINT wu2(0):SOUND
200,.5
828 UL      IF wu3(0)>0 THEN LOCATE 5,57:PRINT wu3(0):SOUND
200,.5
829 kC7      IF z=0 THEN z=1:GOTO wuerfel
830 Vp      wurfw2:
831 rs9      LOCATE 3,68:PRINT wu1(1):SOUND 200,.5

```

```

832 dV      IF wu2(1)>0 THEN LOCATE 4,68:PRINT wu2(1):SOUND
200,.5
833 qg      IF wu3(1)>0 THEN LOCATE 5,68:PRINT wu3(1):SOUND
200,.5
834 9F      z=0:COLOR 1,0
835 pR7      RETURN
836 wn4      auswertung:
837 Y16      IF wu1(0)<=wu1(1) THEN
838 cv8      land(Anr)=land(Anr)-1
839 8r      ELSE
840 GNA      land(Znr)=land(Znr)-1
841 V08      END IF
842 5z6      IF wu2(0)=0 OR wu2(1)=0 THEN GOTO aw2
843 lO      IF wu2(0)<=wu2(1) THEN
844 i18      land(Anr)=land(Anr)-1
845 Ex      ELSE
846 MTA      land(Znr)=land(Znr)-1
847 bU8      END IF
848 HD6      IF wu3(0)=0 OR wu3(1)=0 THEN GOTO aw2
849 yF      IF wu3(0)<=wu3(1) THEN
850 o78      land(Anr)=land(Anr)-1
851 K3      ELSE
852 SZA      land(Znr)=land(Znr)-1
853 ha8      END IF
854 LV6      aw2:
855 OZ8      L=Anr:GOSUB v6
856 T4      f2=f:L=Znr:f=fz:GOSUB v6:f=f2:L=Anr
857 LW      IF land(Znr)<>0 THEN
858 S6A      GOSUB f4:RETURN
859 SB9      ELSE
860 oh      END IF
861 GA      land$(Znr)=Spieler$(spi)
862 kC      land(Znr)=land(Znr)+Wanz:BEEP
863 4U      L=Znr:GOSUB v6:land(Anr)=land(Anr)-Wanz:BEEP
864 yM      L=Anr:GOSUB v6:land$(Znr)=land$(Anr)
865 om      vs(Znr)=1
866 fp      WINDOW OUTPUT 4
867 UbB      IF Karte=0 THEN
868 BKD      RANDOMIZE TIMER
869 JM      z=INT(RND*4)
870 DzE      IF z=0 THEN z=1
871 lV      IF z=1 THEN z$="R"
872 7d      IF z=2 THEN z$="S"
873 mB      IF z=3 THEN z$="K"
874 e7      ka(spi)=ka(spi)+1:k$(spi,ka(spi))=z$
875 9r      Karte=1
876 JS      ELSE
877 5y      END IF
878 W89      RETURN
879 qQ2      verschieben:
880 xO4      CLS:LOCATE 1,1:PRINT "Verschiebe Armeen von "
881 iC      GOSUB welchesLand:vNr=L:vLa$=la$
882 T76      IF vs(vNr)=1 THEN
883 nt8      CLS
884 l5      PRINT "Verschieben erst nach erneutem Angriff möglic
h"
885 oc      GOSUB pause:GOSUB pause:abbruch=1
886 tc      ELSE
887 F8      END IF
888 bk4      IF abbruch=1 THEN abbruch=0:CLS:RETURN
889 Tm      GOSUB eigLand:IF ab=1 THEN ab=0:CLS:RETURN
890 hw5      IF land(vNr)<2 THEN
891 lU7      PRINT "Hier steht nur eine Armee !":GOSUB pause
892 Tc      GOTO verschieben
893 Oj      ELSE
894 MF      END IF
895 K84      LOCATE 1,1:PRINT "Verschiebe Armeen von ";vLa$;" nach "
896 XP      GOSUB welchesLand:nNr=L:nLa$=la$
897 kt      IF abbruch=1 THEN abbruch=0:CLS:RETURN
898 LO      LOCATE 1,1:PRINT "Verschiebe Armeen von ";vLa$;" nach "
;
899 NQ      PRINT nLa$:GOSUB eigLand:IF ab=1 THEN ab=0:CLS:RETURN
900 eb8      FOR p=1 TO 8
901 mg      IF L=Nb(vNr,p) THEN GOTO v2
902 d1      NEXT
903 da      LOCATE 3,5
904 Wx      PRINT "Dies ist kein Nachbarland ! Noch mal.":GOSU
B pause:GOTO verschieben
905 v13      v2:
906 BW4      GOSUB pause
907 7k      WINDOW 4,,(0,200)-(620,235),0,1:GOSUB f4

```

Listing 1. (Fortsetzung)

```

908 TF3   veing:
909 DO4    LOCATE 1,1
910 4G     INPUT "Wieviele Armeen";armeen
911 7t     IF armeen>=land(vNr) OR armeen<=0 THEN CLS:GOTO veing
912 On     land(vNr)=land(vNr)-armeen:L=vNr:GOSUB v6
913 k0     GOSUB f4:vs(nNr)=1
914 wj     land(nNr)=land(nNr)+armeen:L=nNr:GOSUB v6
915 f1     GOSUB f4:CLS:RETURN
916 RA2    eigLand:
917 Yw6    IF land$(L)<>rf$(r) THEN
918 yZ8     LOCATE 3,5:PRINT "Dies ist nicht Dein Land !"
919 7G     GOSUB pause:ab=1:RETURN
920 RA     ELSE
921 ng     END IF
922 Eq7    RETURN
923 Xm2    REM *** Symbolkarten eintauschen ***
924 Dx     tauschen:
925 G14    GOSUB f4:CLS:B=5
926 CG     t2:
927 Aa6    IF ka$(sp1)=0 THEN PRINT "Keine Karten!":RETURN
928 eU     IF rf$(r)>"F" THEN GOTO t3
929 tw     FOR D=1 TO ka$(sp1)
930 387    IF k$(sp1,D)="R" THEN GOSUB Reiter
931 Ra     IF k$(sp1,D)="S" THEN GOSUB Soldat
932 uW     IF k$(sp1,D)="K" THEN GOSUB Kanone
933 YK     B=B+45:NEXT
934 NS6    t3:
935 TV7    rt=0:so=0:k=0:FOR p=1 TO ka$(sp1)
936 RL9    IF k$(sp1,p)="R" THEN rt=rt+1
937 O1     IF k$(sp1,p)="S" THEN so=so+1
938 kS     IF k$(sp1,p)="K" THEN k=k+1
939 QB7    NEXT:wk=1
940 vy9    IF rt>=3 OR so>=3 OR k>=3 THEN GOTO wk
941 4R     IF rt>=1 AND so>=1 AND k>=1 THEN wk=2:GOTO wk
942 YA     RETURN
943 KM8    wk:
944 rA9    IF rf$(r)>"F" THEN GOTO wKarte
945 pA     LOCATE 2,45
946 mc     PRINT "Willst Du tauschen ?";
947 XO     LOCATE 4,62:PRINT "Ja";:LOCATE 4,69:PRINT "Nein";
948 IN     LINE(480,20)-(520,35),,B:LINE(540,20)-(580,35),,B
949 TSA    kt:
950 thC    m=MOUSE(0):IF MOUSE(0)=0 THEN GOTO kt
951 Js     IF MOUSE(3)>480 AND MOUSE(3)<520 THEN GOTO wk
952 RS     arte
953       IF MOUSE(3)>540 AND MOUSE(3)<580 THEN LOCATE
954         5,20:PRINT "ok";
955       CLS:RETURN
956 hH5    wKarte:
957 O16    IF rf$(r)<"G" THEN GOTO wk4
958 KN     FOR D=1 TO ka$(sp1)
959 UZ7    IF k$(sp1,D)="R" THEN GOSUB Reiter
960 s1     IF k$(sp1,D)="S" THEN GOSUB Soldat
961 Lx     IF k$(sp1,D)="K" THEN GOSUB Kanone
962 z1     B=B+45:NEXT
963 Kg     wk4:
964 O69    LOCATE 1,40:PRINT namen$;" tauscht Karten gegen So
965         ldaten !";
966         GOSUB pause
967 m17    IF wk=2 THEN GOTO wk2
968 i19    IF rt>=3 THEN L$="R"
969 zo     IF so>=3 THEN L$="S"
970 xY     IF k>=3 THEN L$="K"
971 1CA    z=0:FOR p=1 TO ka$(sp1):IF z=3 THEN wk3
972 ufC    IF k$(sp1,p)=L$ THEN k$(sp1,p)="+":z=z+1
973 joB    NEXT
974 Ou     GOTO wk3
975 Nh5    wk2:
976 1L7    L$="R":l1$="S":l2$="K":z=0
977 s5     FOR p=1 TO ka$(sp1)
978 169    IF z=3 THEN wk3
979 5o     IF k$(sp1,p)=L$ AND rt<>0 THEN k$(sp1,p)="+":z=z
980       +1:rt=0
981 RA     IF k$(sp1,p)=l1$ AND so<>0 THEN k$(sp1,p)="+":z=
982       z+1:so=0
983 TK     IF k$(sp1,p)=l2$ AND k<>0 THEN k$(sp1,p)="+":z=z
984       +1:k=0
985       NEXT
986       wk3:
987       z=1:FOR p=1 TO ka$(sp1)
988       IF k$(sp1,p)<>"+" THEN K2$(1,z)=k$(sp1,p):z=z+1
989       NEXT:ka$(sp1)=ka$(sp1)-3
990       FOR p=1 TO ka$(sp1)

```

```

985 jxA    k$(sp1,p)=K2$(1,p)
986 z49    NEXT
987 pK     armeen=za:za=za+3:CLS
988 sd     LOCATE 1,1:PRINT namen$;" erhält ";armeen;" Armeen
989 Q18    wk3.3:
990 vB9    IF rf$(r)<"G" THEN
991 BKB     PRINT "Zeige mit der Maus auf eines Deiner Lände
992         r !"
993 K6     GOSUB welchesLand
994 Sd     IF land$(L)<>rf$(r) THEN CLS:GOTO wk3.3
995 I8     land(L)=land(L)+armeen:GOSUB v6
996 x3     GOSUB f4:CLS:RETURN
997 fO     ELSE
998 lu     END IF
999 LS9    GOSUB wKont1:GOSUB pause:CLS
1000 lc    PRINT namen$;" legt seine Armeen auf ";Lname$(L);
1001       "."
1002       GOSUB pause:GOSUB v6
1003       RETURN
1004       Reiter:
1005       LINE(4+B,4)-(41+B,44),,B
1006       LINE(9+B,23)-(12+B,19):LINE-(11+B,18):LINE-(13+B,19)
1007       LINE-(14+B,17):LINE-(14+B,18):LINE-(16+B,22):LINE-(17+B
1008         ,22)
1009       LINE-(11+B,6):LINE-(13+B,7):LINE-(17+B,5):LINE-(15+B,7)
1010       LINE-(17+B,8):LINE-(13+B,9):LINE-(18+B,21):LINE-(19+B,1
1011         9)
1012       LINE-(18+B,17):LINE-(17+B,17):LINE-(18+B,14)
1013       LINE-(20+B,16):LINE-(20+B,18):LINE-(22+B,19)
1014       LINE-(23+B,24):LINE-(25+B,25):LINE-(25+B,27)
1015       LINE-(29+B,28):LINE-(32+B,28):LINE-(38+B,30)
1016       LINE-(36+B,30):LINE-(32+B,31):LINE-(30+B,30)
1017       LINE-(23+B,40):LINE-(22+B,42):LINE-(20+B,42)
1018       LINE-(21+B,41):LINE-(22+B,40):LINE-(24+B,33)
1019       LINE-(19+B,38):LINE-(19+B,39):LINE-(18+B,40)
1020       LINE-(18+B,37):LINE-(22+B,32):LINE-(20+B,32)
1021       LINE-(20+B,34):LINE-(18+B,34):LINE-(18+B,32)
1022       LINE-(17+B,31):LINE-(14+B,35):LINE-(16+B,37)
1023       LINE-(16+B,39):LINE-(13+B,36):LINE-(13+B,32)
1024       LINE-(10+B,33):LINE-(8+B,37):LINE-(7+B,35):LINE-(10+B,3
1025         1)
1026       LINE-(14+B,30):LINE-(12+B,23):LINE-(10+B,25)
1027       LINE-(9+B,24):LINE-(8+B,25):LINE-(9+B,23)
1028       PAINT(18+B,28),f,1
1029       RETURN
1030       Soldat:
1031       LINE(4+B,4)-(41+B,44),,B
1032       LINE(14+B,44)-(16+B,43):LINE-(17+B,36):LINE-(16+B,29)
1033       LINE-(15+B,24):LINE-(14+B,24):LINE-(14+B,22):LINE-(15+B
1034         ,22)
1035       LINE-(14+B,17):LINE-(15+B,17):LINE-(13+B,10):LINE-(15+B
1036         ,17)
1037       LINE-(16+B,22):LINE-(17+B,22):LINE-(17+B,17):LINE-(19+B
1038         ,17)
1039       LINE-(19+B,15):LINE-(17+B,14):LINE-(18+B,12):LINE-(17+B
1040         ,12)
1041       LINE-(17+B,10):LINE-(24+B,10):LINE-(23+B,13):LINE-(23+B
1042         ,16)
1043       LINE-(25+B,14):LINE-(26+B,14):LINE-(26+B,16):LINE-(27+B
1044         ,17)
1045       LINE-(28+B,22):LINE-(26+B,24):LINE-(28+B,28):LINE-(26+B
1046         ,30)
1047       LINE-(28+B,34):LINE-(26+B,33):LINE-(25+B,41):LINE-(26+B
1048         ,44)
1049       LINE-(14+B,44):LINE(16+B,24)-(18+B,24):LINE-(18+B,27)
1050       LINE-(16+B,24):LINE(20+B,43)-(22+B,35):LINE-(23+B,43):L
1051         INE-(20+B,43)
1052       PAINT(20+B,30),f,1:RETURN
1053       Kanone:
1054       LINE(2+B,4)-(40+B,44),,B
1055       LINE(6+B,18)-(8+B,19):LINE-(19+B,19):LINE-(19+B,17)
1056       LINE-(23+B,17):LINE-(23+B,19):LINE-(31+B,17):LINE-(31+B
1057         ,19)
1058       LINE-(33+B,19):LINE-(33+B,21):LINE-(31+B,21):LINE-(31+B
1059         ,22)
1060       LINE-(29+B,22):LINE-(30+B,24):LINE-(29+B,25):LINE-(34+B
1061         ,29)
1062       LINE-(39+B,28):LINE-(36+B,30):LINE-(37+B,31):LINE-(39+B
1063         ,31)
1064       LINE-(39+B,35):LINE-(10+B,35):LINE-(13+B,32):LINE-(19+B
1065         ,31)
1066       LINE-(16+B,29):LINE-(15+B,27):LINE-(15+B,25):LINE-(15+B

```



```

,23)
1048 X5 LINE-(14+B,23):LINE-(13+B,22):LINE-(8+B,22):LINE-(6+B,2
3)
1049 1s LINE-(6+B,18)
1050 Ev LINE(17+B,22)-(20+B,24):LINE-(17+B,24):LINE-(17+B,22)
1051 sm LINE(17+B,26)-(20+B,25):LINE-(18+B,27):LINE-(17+B,26)
1052 r1 LINE(19+B,29)-(21+B,26):LINE-(21+B,30):LINE-(19+B,29)
1053 11 LINE(23+B,30)-(23+B,26):LINE-(26+B,29):LINE-(23+B,30)
1054 42 LINE(24+B,25)-(26+B,28):LINE-(28+B,27):LINE-(24+B,25)
1055 YL LINE(26+B,22)-(27+B,22):LINE-(27+B,24):LINE-(26+B,22)
1056 X0 PAINT(14+B,34),f,1
1057 P1 RETURN
1058 aX2 nextSpieler:
1059 Ja4 r=r+1:IF r>spges THEN r=1
1060 OJ6 g=0:FOR p=1 TO 42
1061 gZ8 IF land$(p)=rf$(r) THEN g=1
1062 T1 vs(p)=0
1063 yP6 NEXT:IF g=0 AND rf$(r)>"F" THEN csp=csp-1
1064 64 IF g=0 THEN rf$(r)="*"
1065 4U4 IF rf$(r)="*" THEN GOTO nextSpieler
1066 eP Karte=0:GOTO beginn
1067 7D0 sieg:
1068 v52 WINDOW OUTPUT 4
1069 nt CLS
1070 kr PRINT namen$;" ist der Sieger"
1071 OW GOSUB pause:MENU RESET
1072 2x END
1073 D50 f4:
1074 pm2 WINDOW OUTPUT 4:RETURN
1075 oi ende:
1076 em4 MENU RESET:STOP
1077 1t1 REM **** Der Computer zieht ****
1078 S4 computer:
1079 PM3 IF za>21 THEN dugmax=10
1080 N2 IF za>33 THEN dugmax=15
1081 y1 GOSUB wKonti
1082 cv5 PRINT namen$;" legt seine Armeen auf ";Lname$(L)
1083 283 GOSUB v6:GOSUB pause
1084 xG5 dug=dug+(.8/csp):angriff=0:angri=0
1085 Bg3 comangr:
1086 sz7 ON MENU GOSUB ende:MENU ON
1087 16 IF dug>dugmax THEN dug=dugmax
1088 j1 IF ka(spi)>2 THEN GOSUB tauschen
1089 Ju maxi=-100:max=-100:angriff=0:anz=0
1090 Ch FOR L=1 TO 42
1091 Ao9 IF land$(L)=rf$(r) AND land(L)>1 THEN GOSUB ang
1092 hm7 NEXT
1093 up6 IF maxi<2 THEN angri=0:GOSUB cver:GOTO ang2
1094 m1 IF angri=1 THEN GOTO ang1
1095 nJ IF angri=0 THEN GOTO ang2
1096 E6 ang:
1097 EW8 FOR N=1 TO 10
1098 SN Nbar=Nb(L,N):IF Nbar=0 THEN RETURN
1099 aJ IF land$(Nbar)<>rf$(r) THEN max=land(L)-land(Nbar)
):angriff=1
IF land$(Nbar)<"G" THEN max=max+.5
IF max>max1 THEN
max1=max:Anr=L:Znr=Nbar
FOR p=1 TO spges
IF land$(Znr)=Spieler$(p) THEN fz=farb(p)
NEXT p
ELSE
END IF
NEXT
ang1:
Ausgl$=Lname$(Anr):Ziell$=Lname$(Znr)
FOR p=1 TO spges
IF land$(Znr)=Spieler$(p) THEN Znamen$=N$(p)
NEXT
GOSUB f4:CLS
PRINT "Von ";Ausgl$;" nach ";Ziell$
GOSUB pause
z=0:GOSUB wuerfel:GOSUB pause:GOSUB auswertung
angri=angri+.5:GOSUB pause
ang2:
IF angri>dug OR angri=0 THEN
angriff=0:anz=0:GOTO nextSpieler
ELSE
END IF
GOTO comangr
pause:
FOR p=1 TO 3000:NEXT:RETURN
1127 Jd0 REM *** Computer verschiebt Armeen ***

```

```

1128 6K2 cver:
1129 pK4 FOR L=1 TO 42
1130 zF6 IF land$(L)=rf$(r) AND land(L)>3 THEN GOSUB cverp
1131 KP4 NEXT
1132 cE6 RETURN
1133 FP3 cverp:
1134 4C5 S=0:FOR p=1 TO 8:N(p)=Nb(L,p)
1135 w99 IF N(p)=0 THEN GOTO cv1
1136 nM7 IF land$(N(p))<>rf$(r) THEN S=1
1137 r15 cv1:
1138 qK NEXT:IF S=1 THEN angri=0:RETURN
1139 F1 armeen=land(L)-2
1140 f16 n2=0:FOR p=1 TO 8:FOR D=1 TO 8
1141 kr8 n1=Nb(N(p),D)
1142 fR9 IF n1=0 THEN GOTO cv2
1143 O18 IF land$(n1)<>rf$(r) THEN S=1:n2=N(p)
1144 2D6 cv2:
1145 Pz NEXT:NEXT:IF n2=0 THEN angri=0:RETURN
1146 PP5 GOSUB f4:CLS:PRINT namen$;" verschiebt von ";
1147 vp6 PRINT Lname$(L);" nach ";Lname$(n2);" ";
1148 fZ PRINT armeen;" Armeen."
1149 6R8 GOSUB pause
1150 sk land(L)=land(L)-armeen:GOSUB v6
1151 LI land(n2)=land(n2)+armeen:L=n2:GOSUB v6
1152 S6 GOSUB pause:angriff=1
1153 xZ6 RETURN
1154 dR3 wKonti:
1155 2O5 conti=0:contimax=16:FOR L=1 TO 9
1156 3g IF land$(L)=rf$(r) THEN conti=conti+1
1157 kp NEXT
1158 4H2 IF conti=0 THEN GOTO Samerika
1159 RF9 IF land$(27)=rf$(r) THEN conti=conti+.5
1160 8t IF land$(10)=rf$(r) THEN conti=conti+.5
1161 H6 IF land$(14)=rf$(r) THEN conti=conti+.5
1162 p5 IF conti=10.5 THEN contimax=15:conti=0:GOTO Sameri
ka
1163 Y1 contimax=(10.5/conti):conti=0:l1=1:l2=L-1
1164 J64 Samerika:
1165 k45 FOR L=10 TO 13
1166 Dq6 IF land$(L)=rf$(r) THEN conti=conti+1
1167 uz5 NEXT
1168 gO7 IF conti=0 THEN GOTO Europa
1169 dm9 IF land$(9)=rf$(r) THEN conti=conti+.5
1170 1S IF land$(37)=rf$(r) THEN conti=conti+.5
1171 C1 contimax2=5/conti
1172 Ho GOSUB conti2
1173 SS IF contimax2<contimax THEN contimax=contimax2:l1=
10:l2=L-1
1174 VU conti=0
1175 1t4 Europa:
1176 OM6 FOR L=14 TO 20
1177 O17 IF land$(L)=rf$(r) THEN conti=conti+1
1178 5A6 NEXT
1179 dH8 IF conti=0 THEN GOTO Asien
1180 E1A IF land$(3)=rf$(r) THEN conti=conti+.5
1181 bJ IF land$(21)=rf$(r) THEN conti=conti+.5
1182 eN IF land$(22)=rf$(r) THEN conti=conti+.5
1183 hR IF land$(23)=rf$(r) THEN conti=conti+.5
1184 wG IF land$(37)=rf$(r) THEN conti=conti+.5
1185 zk IF land$(38)=rf$(r) THEN conti=conti+.5
1186 d0 contimax2=10/conti:GOSUB conti2
1187 1R IF contimax2<contimax THEN contimax=contimax2:l1
=14:l2=20
1188 j1 conti=0
1189 vT4 Asien:
1190 Jg6 FOR L=21 TO 32
1191 oF7 IF land$(L)=rf$(r) THEN conti=conti+1
1192 JO6 NEXT
1193 wV8 IF conti=0 THEN GOTO Austra
1194 ysA IF land$(19)=rf$(r) THEN conti=conti+.5
1195 xq IF land$(18)=rf$(r) THEN conti=conti+.5
1196 Av IF land$(38)=rf$(r) THEN conti=conti+.5
1197 Dz IF land$(39)=rf$(r) THEN conti=conti+.5
1198 21 IF land$(33)=rf$(r) THEN conti=conti+.5
1199 5m IF land$(34)=rf$(r) THEN conti=conti+.5
1200 M1 IF land$(1)=rf$(r) THEN conti=conti+.5
1201 O1 contimax2=15.5/conti:GOSUB conti2
1202 Ok IF contimax2<contimax THEN contimax=contimax2:l1
=21:l2=32
1203 yx conti=0
1204 1n5 Austra:

```

Listing 1. (Fortsetzung)

```

1205 3X7      FOR L=33 TO 36
1206 rU9      IF land$(L)=rf$(r) THEN conti=conti+1
1207 Yd7      NEXT
1208 Qk9      IF conti=0 THEN GOTO Afrika
1209 9nB      IF land$(31)=rf$(r) THEN conti=conti+.5
1210 pw      contimax2=4.5/conti:GOSUB conti2
1211 tH      IF contimax2<contimax THEN contimax=contimax2:1
1212 76      1=33:12=36
1213 ob5      conti=0
1214 Ch7      Afrika:
1215 Od9      FOR L=37 TO 42
1216 hm7      IF land$(L)=rf$(r) THEN conti=conti+1
1217 ZK9      NEXT
1218 8vB      IF conti=0 THEN GOTO kontiend
1219 JB      IF land$(12)=rf$(r) THEN conti=conti+.5
1220 MF      IF land$(17)=rf$(r) THEN conti=conti+.5
1221 J3      IF land$(18)=rf$(r) THEN conti=conti+.5
1222 kR      IF land$(23)=rf$(r) THEN conti=conti+.5
1223 JS      contimax2=8/conti:GOSUB conti2
1224 JI      IF contimax2<contimax THEN contimax=contimax2:1
1225 Md5      1=37:12=42
1226 TG7      conti=0
1227 vU      kontiend:
1228 he9      max=0:maxi=0:start=0
1229 fn7      FOR L=11 TO 12:p=0
1230 Fm9      IF land$(L)<>rf$(r) THEN GOTO p2
1231 pe      p:
1232 tz      p=p+1:nbl=Nb(L,p)
1233 Jt      IF nbl=0 THEN GOTO p2
1234 1M      IF land$(nbl)=rf$(r) THEN GOTO p
1235 BS      IF land$(nbl)<>rf$(r) AND nbl>11-1 AND nbl<12+
1236 447      1 THEN max=max+3:IF max>maxi THEN maxi=max:13=L:
1237 9b9      start=1:GOTO p
1238 Ux6      IF land$(nbl)<>rf$(r) THEN max=max+1:IF max>max
1239 9d8      i THEN maxi=max:13=L:start=1
1240 wu      GOTO p
1241 IL7      p2:
1242 Ry8      max=0:NEXT
1243 Jt      IF start=1 THEN start=0:GOTO gefunden
1244 9k      FOR L=1 TO 42:p=0
1245 St      IF land$(L)<>rf$(r) THEN GOTO p3
1246 HI7      p3:
1247 CH9      p5:
1248 MO7      p=p+1:nbl=Nb(L,p)
1249 mG9      IF nbl=0 THEN GOTO p3
1250 Nz6      IF land$(nbl)<>rf$(r) THEN land(13)=land(13)+arnee
1251 FN8      en:GOTO p4
1252 YA      IF land$(nbl)=rf$(r) THEN GOTO p5
1253 us6      p3:
1254 yN8      NEXT
1255 bD      p4:
1256 v92      L=13:start=0:RETURN
1257 az4      gefunden:
1258 Kw      land(13)=land(13)+arnee:L=13
1259 mG      RETURN
1260 H1      conti2:
1261 kQ      IF contimax2=1 THEN contimax2=17
1262 LO      RETURN
1263 mD      mausposition:
1264 rw      DATA 15,70,0,40,"Alaska"
1265 dP      DATA 78,175,0,35,"Nord-West-Territorium"
1266 rs      DATA 180,290,0,33,"Grönland"
1267 Py      DATA 75,160,37,54,"Alberta"
1268 Gs      DATA 163,205,36,55,"Ontario"
1269 Vb      DATA 212,245,41,59,"Quebeck"
1270 Oq      DATA 71,141,57,78,"Weststaaten"
1271 UO      DATA 143,216,57,89,"Oststaaten"
1272 Mr      DATA 70,135,79,104,"Mittelamerika"
1273 5N      DATA 64,175,104,122,"Venezuela"
1274 1H      DATA 56,116,123,162,"Peru"
1275 Jy      DATA 107,213,123,167,"Brasilien"
1276 ZC      DATA 71,136,160,220,"Argentinien"
1277 aN      DATA 250,303,43,61,"Island"
1278 VG      DATA 305,352,36,73,"Skandinavien"
1279 zs      DATA 290,360,77,91,"Mitteleuropa"
1280 hb      DATA 266,312,89,109,"Westeuropa"
1281 YL      DATA 316,367,91,108,"Südeuropa"
1282 uL      DATA 355,424,18,100,"Ukraine"
1283 GM      DATA 234,280,74,96,"Groß Britannien"
1284 rn      DATA 430,493,4,70,"Ural"
1285 Bk      DATA 427,478,71,110,"Afghanistan"
1286 Jv      DATA 377,446,104,126,"Mittlerer Osten"
1287 cG      DATA 496,533,5,78,"Sibirien"
1288 xk      DATA 540,587,5,28,"Jakutien"
1289 NX      DATA 532,560,29,75,"Irkutsk"

```

```

1283 GM      DATA 575,615,0,79,"Kamtschatka"
1284 rn      DATA 517,561,75,91,"Mongolei"
1285 Bk      DATA 479,566,85,110,"China"
1286 Jv      DATA 449,511,112,141,"Indien"
1287 cG      DATA 513,560,116,140,"Siam"
1288 xk      DATA 569,600,80,110,"Japan"
1289 NX      DATA 575,620,135,160,"Neu Guinea"
1290 Sq      DATA 510,575,141,163,"Indonesien"
1291 oS      DATA 518,570,163,200,"West-Australien"
1292 GU      DATA 575,620,160,195,"Ost-Australien"
1293 cN      DATA 245,300,110,156,"Nord-west-Afrika"
1294 h8      DATA 305,380,112,131,"Ägypten"
1295 qW      DATA 356,405,132,167,"Ostafrika"
1296 O5      DATA 320,357,146,164,"Kongo"
1297 Oc      DATA 305,380,168,210,"Südafrika"
1298 ZO      DATA 390,420,169,200,"Madagaskar"
1299 FR      REM gespeichert in Nb
1300 KE2      Nachbarland:
1301 hO4      DATA 2,4,27,0,0,0,0,0
1302 MW      DATA 1,3,5,4,0,0,0,0
1303 Kk      DATA 2,5,6,14,0,0,0,0
1304 Tf      DATA 1,2,5,7,0,0,0,0
1305 M1      DATA 2,3,6,4,7,8,0,0
1306 Ig      DATA 3,5,8,0,0,0,0,0
1307 CK      DATA 4,5,8,9,0,0,0,0
1308 4S      DATA 9,7,5,6,0,0,0,0
1309 v1      DATA 7,8,10,0,0,0,0,0
1310 nJ      DATA 11,12,9,0,0,0,0,0
1311 bh      DATA 10,12,13,0,0,0,0,0
1312 OP      DATA 10,11,13,37,0,0,0,0
1313 PC      DATA 11,12,0,0,0,0,0,0
1314 eh      DATA 15,20,3,0,0,0,0,0
1315 g3      DATA 14,20,16,19,0,0,0,0
1316 tJ      DATA 15,20,19,18,17,0,0,0
1317 J6      DATA 20,16,18,37,0,0,0,0
1318 UW      DATA 17,16,19,37,38,23,0,0
1319 TJ      DATA 15,16,18,23,22,21,0,0
1320 66      DATA 14,15,16,17,0,0,0,0
1321 VG      DATA 22,29,24,19,0,0,0,0
1322 JI      DATA 21,29,30,23,19,0,0,0
1323 6k      DATA 22,30,19,18,38,39,0,0
1324 ek      DATA 21,29,28,26,25,0,0,0
1325 pq      DATA 24,26,27,0,0,0,0,0
1326 Xd      DATA 25,24,28,27,0,0,0,0
1327 ka      DATA 25,26,28,32,1,0,0,0
1328 N1      DATA 27,26,24,29,32,0,0,0
1329 fS      DATA 28,24,21,22,30,31,32,0
1330 HK      DATA 23,22,29,31,0,0,0,0
1331 ue      DATA 30,29,34,0,0,0,0,0
1332 PN      DATA 27,28,29,0,0,0,0,0
1333 U8      DATA 34,36,0,0,0,0,0,0
1334 bS      DATA 33,35,36,31,0,0,0,0
1335 WA      DATA 34,36,0,0,0,0,0,0
1336 pt      DATA 33,35,34,0,0,0,0,0
1337 dW      DATA 40,39,38,17,18,12,0,0
1338 2s      DATA 37,39,23,18,0,0,0,0
1339 Fx      DATA 38,37,40,41,42,23,0,0
1340 KA      DATA 37,39,41,0,0,0,0,0
1341 6r      DATA 40,39,42,0,0,0,0,0
1342 TX      DATA 39,41,0,0,0,0,0,0

```

(C) 1987 M&T

Listing 1. (Schluß)

Programmname:	Lader
Computer:	A 500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2
Bemerkung:	Lädt Listing 1 nach, das unter dem Namen »Hauptprogramm« gespeichert sein muß

Programm : Lader

```

1 NJO REM Basic-Speicher vergrößern
2 Dk CLEAR,40000&
3 O3 CHAIN "Hauptprogramm"

```

(C) 1987 M&T

Listing 2. Das Ladeprogramm erweitert den Basic-Speicher. Bitte mit dem Checksummer (Seite 159) eingeben.

Fortsetzung von Seite 22

Finanzen im Griff

```

20 y4 farbe0=0
21 5D farbe1=1
22 CM farbe2=2
23 JV farbe3=3
24 Qe farbe4=4
25 Xn farbe5=5
26 ew farbe6=6
27 15 farbe7=7
28 xF WINDOW OUTPUT 2
29 HO COLOR farbe6,0
30 2D LOCATE 1,1
31 sh PRINT "Zeit";
32 8K LOCATE 2,1
33 SP PRINT "Datum";
34 ER LOCATE 3,1
35 g3 PRINT "Bezugszeit";
36 KY LOCATE 4,1
37 4D PRINT "Kontoname";
38 sB LOCATE 1,30
39 01 PRINT "Arbeitsdauer";
40 yI LOCATE 2,30
41 sL PRINT "Buchungen";
42 4P LOCATE 3,30
43 gI PRINT "Speicher";
44 7T WINDOW OUTPUT 1
45 81 ON ERROR GOTO fehler
46 0d ON BREAK GOSUB unterbrechung
47 jw BREAK ON
48 mr ON MENU GOSUB wahl
49 oM ON TIMER(1) GOSUB uhr
50 RX TIMER ON
51 3T arbeitsdauer=TIMER
52 1a GOSUB aktuell
53 4N GOSUB pruefung
54 6W MENU ON
55 2Y BEEP
56 DO WHILE 1
57 1L SLEEP
58 NB WEND
59 CM wahl:
60 Ks v=MENU(0):w=MENU(1)
61 HB MENU OFF
62 Tk ON v GOSUB haupt1,haupt2,haupt3,haupt4,haupt5,haupt6,haupt7
63 C1 GOSUB aktuell
64 Fa weiter:
65 Hh MENU ON
66 Q2 RETURN
67 QZ haupt1:
68 ru konto$=verzeichnis$(w):kontonr=w
69 T5 RETURN
70 X1 haupt3:
71 Rt ON w GOTO menu31,menu32,menu33
72 Xh haupt2:
73 6Q ON w GOTO menu21,menu22,menu23,menu25,menu26,menu27,menu28,
    menu29
74 dp haupt4:
75 Nj ON w GOTO menu41,menu42
76 hu haupt5:
77 zN ON w GOTO menu51,menu52,menu53
78 lz haupt6:
79 Fb ON w GOTO menu61,menu62,menu63
80 p4 haupt7:
81 ZJ ON w GOTO menu71,menu72,menu73,menu74,menu75,menu76
82 a6 menu21:
83 yv IF konto$="" THEN CALL meldung("Kein Konto definiert"):RETU
    RN
84 E4 IF index=99 THEN CALL meldung("Buchungen nur nach Neustart
    möglich"):RETURN
85 dJ CALL requester("Um welche Art Buchung handelt es sich","Ei
    nnahme","Ausgabe")
86 60 IF req THEN
87 LZ vorzeichen=1
88 1k ELSE
89 H1 vorzeichen=-1
90 OH END IF
91 MK ein:
92 rV WINDOW 5,,(0,100)-(maxbreite,135),0,1
93 nS LOCATE 2,2:COLOR farbe5,0:PRINT "Bezeichnung"
94 00 LOCATE 4,2:PRINT "Betrag";

```

```

95 uR LOCATE 2,40:COLOR farbe5,0:PRINT STRING$(20,"_");STRING$(2
    0," ")
96 ZI LOCATE 2,40:COLOR farbe1,0:INPUT a$:IF LEN(a$)>18 THEN BEE
    P:GOTO ein
97 Be IF a$="" THEN menu211
98 GP ein1:
99 My LOCATE 4,40:COLOR farbe5,0:PRINT STRING$(14,"_");STRING$(2
    0," ")
100 8J LOCATE 4,40:COLOR farbe1,0:INPUT b$
101 ce IF VAL(b$)>99999999.99# THEN BEEP:GOTO ein1
102 OM IF VAL(b$)=0 THEN menu211
103 Om index=index+1
104 kf nummer(index)=kontonr
105 e6 konto$(index)=konto$
106 p5 jahr$(index)=jahr$
107 4Z monat$(index)=monat$
108 En tag$(index)=tag$
109 gv buchung$(index)=UCASE$(a$)
110 Tb betrag$(index)=STR$(VAL(b$)*vorzeichen)
111 ce IF VAL(b$)>99999999.99# THEN BEEP:GOTO ein1
112 OM IF VAL(b$)=0 THEN menu211
113 Om index=index+1
114 kf nummer(index)=kontonr
115 e6 konto$(index)=konto$
116 p5 jahr$(index)=jahr$
117 4Z monat$(index)=monat$
118 En tag$(index)=tag$
119 gv buchung$(index)=UCASE$(a$)
120 Tb betrag$(index)=STR$(VAL(b$)*vorzeichen)
121 I5 menu211:
122 Fy WINDOW CLOSE 5
123 Ea WINDOW OUTPUT 1
124 Co RETURN
125 9g menu22:
126 Bn WINDOW 5,,(0,100)-(maxbreite,115),0,1
127 Ux LOCATE 2,2:COLOR farbe5,0:PRINT "Bitte neuen Kontonamen ein
    geben"
128 20 LOCATE 2,40:COLOR farbe5,0:PRINT STRING$(20,"_")
129 R9 LOCATE 2,40:COLOR farbe1,0:INPUT a$:IF LEN(a$)>18 THEN BEE
    P:GOTO menu22:
130 e3 IF a$="" THEN menu221
131 fb a$=UCASE$(a$)
132 Rf CALL requester("Kontonamen "+a$+" speichern?","Ja","Nein"
    )
133 hz IF req THEN
134 k04 konto$=a$
135 DQ maxxkonto=maxkonto+1
136 K7 verzeichnis$(maxkonto)=a$
137 fD flag$(maxkonto)=1
138 N2 Z=0
139 Zv IF maxxkonto<>0 THEN
140 mp FOR i=1 TO maxxkonto
141 jK IF flag$(i)=0 THEN Z=1
142 Zp NEXT i
143 vf IF Z<>0 THEN
144 OJ SWAP verzeichnis$(maxkonto),verzeichnis$(Z)
145 9H SWAP flag$(maxkonto),flag$(Z)
146 W1 maxxkonto=maxkonto-1
147 92 END IF
148 A3 END IF
149 NW OPEN "0",#1,verzeichnis$+jahr$
150 wz FOR i=1 TO maxxkonto
151 L4 PRINT #1,verzeichnis$(i)
152 m1 PRINT #1,flag$(i)
153 zX MENU 1,i,flag$(i),verzeichnis$(i)
154 L1 NEXT i
155 ZM CLOSE #1
156 Q1 GOSUB einlesen
157 JCO END IF
158 vj menu221:
159 qZ WINDOW CLOSE 5
160 pB WINDOW OUTPUT 1
161 nP RETURN
162 mK menu23:
163 63 IF konto$="" THEN CALL meldung("Kein Konto definiert"):RETU
    RN
164 8y CALL requester("Soll Konto "+konto$+" gelöscht werden?","J
    a","Nein")
165 DV IF req THEN
166 en OPEN "0",#1,verzeichnis$+jahr$
167 DG FOR i=1 TO maxxkonto
168 jF IF konto$<>verzeichnis$(i) THEN

```

Listing 1. (Fortsetzung)

```

159 dM PRINT #1,verzeichnis$(i)
160 40 PRINT #1,flag$(i)
161 Cv ELSE
162 nx PRINT #1,verzeichnis$(i):flag$(i)=0
163 kr PRINT #1,0
164 sS MENU 1,1,0,verzeichnis$(i)
165 bU END IF
166 7N NEXT i
167 vI CLOSE #1
168 DO OPEN "R", #2,jahr$,8
169 UT FIELD #2,8 AS x$
170 wG LSET x$=MKD$(0)
171 c2 PUT #2,kontonr
172 4s CLOSE #2
173 UM OPEN "R", #2,jahr$+monat$,8
174 ZY FIELD #2,8 AS x$
175 1L LSET x$=MKD$(0)
176 h7 PUT #2,kontonr
177 9x CLOSE #2
178 Oz konto$=""
179 pI END IF
180 Gs RETURN
181 Jt menu25:
182 7d summe#=0
183 aX IF konto$="" THEN CALL meldung("Kein Konto definiert"):RETU
RN
184 A8 OPEN "I", #1,konto$+jahr$+monat$
185 ex zeile=4
186 RR WHILE NOT EOF(1)
187 PU IF zeile=4 THEN
188 ag CLS
189 Wz LINE (0,10)-(520,10),farbe4
190 Zp LINE (0,25)-(520,25),farbe4
191 UB LINE (520,0)-(520,200),farbe4
192 Pb LINE (0,0)-(0,200),farbe4
193 yI COLOR farbe7,0
194 7H LOCATE 1,2:PRINT "Einzelkonto";TAB(45);"Jahr";TAB(55);"Mona
t";
195 kH LOCATE 3,2:PRINT "Tag"
196 m9 LOCATE 3,12:PRINT "Monat"
197 9P LOCATE 3,22:PRINT "Jahr"
198 fq LOCATE 3,32:PRINT "Bezeichnung"
199 QJ LOCATE 3,52:PRINT "Betrag"
200 2I COLOR farbe6,0
201 Tz LOCATE 1,15
202 qE PRINT konto$;TAB(50);jahr$;TAB(60)monat$;
203 we COLOR farbe3,0
204 E7 END IF
205 uI zeile=zeile+1
206 x3 INPUT #1,a$,b$,c$,d$,e
207 d7 LOCATE zeile,2
208 Ob PRINT c$;
209 Rb LOCATE zeile,13:PRINT b$;
210 Nb LOCATE zeile,22:PRINT a$;
211 fx LOCATE zeile,32:PRINT d$;
212 qt LOCATE zeile,51:PRINT USING "#####.##";e;
213 cd LOCATE zeile+1,1:PRINT PTAB(1);SPACE$(63);
214 5M summe#=summe#+e:LOCATE zeile+2,45
215 uS LINE (1,(zeile+1)*8)-(519,(zeile+1)*8),farbe0
216 Qk LINE (1,(zeile+2)*8)-(519,(zeile+2)*8),farbe4
217 Ox LINE (1,(zeile)*8-1)-(519,(zeile)*8-1),farbe0
218 Iz LINE (1,(zeile+1)*8-1)-(519,(zeile+1)*8-1),farbe4
219 Om PRINT USING "Saldo:#####.##";summe#
220 W7 IF zeile=23 THEN zeile=4:CALL meldung("Maustaste")
221 Oo WEND
222 ob CLOSE #1
223 zX CALL meldung("Maustaste")
224 Bp COLOR farbe1,0
225 BH CLS
226 Oc RETURN
227 5g menu26:
228 rN summe#=0
229 MD OPEN "R", #1,jahr$+monat$,8
230 pS FIELD #1,8 AS ein$
231 HN CLS
232 dg LINE (0,10)-(380,10),farbe4
233 gW LINE (0,25)-(380,25),farbe4
234 TS LINE (380,0)-(380,200),farbe4
235 6I LINE (0,0)-(0,200),farbe4
236 fP COLOR farbe7,0
237 oT LOCATE 1,2:PRINT "Monatskonto";TAB(30);"Jahr";TAB(40);"Mona
t";
238 Wk LOCATE 3,2:PRINT "Konto"
239 qG LOCATE 3,31:PRINT "Betrag"
240 gP COLOR farbe6,0

```

```

241 Jh LOCATE 1,35
242 tP PRINT jahr$;TAB(46)monat$;
243 aG COLOR farbe3,0
244 cf FOR i=1 TO maxkonto
245 VH LSET ein$=""
246 V1 GET #1,1
247 cy LOCATE i+4,2
248 wQ PRINT verzeichnis$(i),
249 3J LOCATE i+4,30:PRINT USING "#####.##";CVD
(ein$);
250 yL LOCATE i+5,1:PRINT PTAB(1);SPACE$(45);
251 GT summe#=summe#+CVD(ein$):LOCATE i+6,24
252 eP LINE (1,(i+5)*8)-(379,(i+5)*8),farbe0
253 wn LINE (1,(i+6)*8)-(379,(i+6)*8),farbe4
254 F6 LINE (1,(i+4)*8-1)-(379,(i+4)*8-1),farbe0
255 VS LINE (1,(i+5)*8-1)-(379,(i+5)*8-1),farbe4
256 zN PRINT USING "Saldo:#####.##";summe#
257 aq NEXT i
258 OB CLOSE #1
259 Z7 CALL meldung("Maustaste")
260 lP COLOR farbe1,0
261 lr CLS
262 aC RETURN
263 hJ menu27:
264 Rx summe#=0
265 kW OPEN "R", #1,jahr$,8
266 P2 FIELD #1,8 AS ein$
267 rx CLS
268 DG LINE (0,10)-(380,10),farbe4
269 G6 LINE (0,25)-(380,25),farbe4
270 32 LINE (380,0)-(380,200),farbe4
271 gs LINE (0,0)-(0,200),farbe4
272 Fz COLOR farbe7,0
273 sX LOCATE 1,2:PRINT "Jahreskonto";TAB(30);"Jahr";
274 6K LOCATE 3,2:PRINT "Konto"
275 Qq LOCATE 3,31:PRINT "Betrag"
276 Gz COLOR farbe6,0
277 tH LOCATE 1,35
278 n2 PRINT jahr$;
279 Aq COLOR farbe3,0
280 CF FOR i=1 TO maxkonto
281 5r LSET ein$=""
282 5b GET #1,1
283 CY LOCATE i+4,2
284 Fy PRINT verzeichnis$(i);
285 dt LOCATE i+4,30:PRINT USING "#####.##";CVD
(ein$);
286 Yv LOCATE i+5,1:PRINT PTAB(1);SPACE$(45);
287 q3 summe#=summe#+CVD(ein$):LOCATE i+6,24
288 Ez LINE (1,(i+5)*8)-(379,(i+5)*8),farbe0
289 WN LINE (1,(i+6)*8)-(379,(i+6)*8),farbe4
290 pg LINE (1,(i+4)*8-1)-(379,(i+4)*8-1),farbe0
291 52 LINE (1,(i+5)*8-1)-(379,(i+5)*8-1),farbe4
292 Zx PRINT USING "Saldo:#####.##";summe#
293 AQ NEXT i
294 yI CLOSE #1
295 9h CALL meldung("Maustaste")
296 Lz COLOR farbe1,0
297 LR CLS
298 Am RETURN
299 Jw menu28:
300 3I IF index=0 THEN CALL meldung("Buchungsspeicher leer"):RETUR
N
301 su zaehler=0
302 au zeile=5
303 Dz WHILE zaehler<index
304 TH zaehler=zaehler+1
305 PQ IF zeile=5 THEN
306 Ua CLS
307 K5 LINE (0,18)-(570,18),farbe4
308 ko LINE (0,33)-(570,33),farbe4
309 sj LINE (570,0)-(570,200),farbe4
310 JV LINE (0,0)-(0,200),farbe4
311 sc COLOR farbe7,0
312 oJ LOCATE 2,22:PRINT "Aktuelle Buchungen:"
313 Pu LOCATE 4,2:PRINT "Konto";
314 sb LOCATE 4,22:PRINT "Tag";
315 rg LOCATE 4,26:PRINT "Monat";
316 ER LOCATE 4,32:PRINT "Jahr"
317 q8 LOCATE 4,38:PRINT "Bezeichnung"
318 d4 LOCATE 4,59:PRINT "Betrag"
319 uc COLOR farbe5,0
320 6z END IF
321 mA zeile=zeile+1
322 lP LOCATE zeile,2:PRINT konto$(zaehler);

```



```

323 eF LOCATE zeile,22:PRINT tag$(zaehler);
324 wk LOCATE zeile,27:PRINT monat$(zaehler);
325 mv LOCATE zeile,32:PRINT jahr$(zaehler);
326 i9 LOCATE zeile,38:PRINT buchung$(zaehler);
327 j9 LOCATE zeile,59:PRINT USING "###.###.###.###";
    VAL(betrag$(zaehler));
328 lx IF zeile=23 THEN zeile=5:CALL meldung ("Maustaste")
329 ky WEND
330 iG CALL meldung ("Maustaste")
331 uy COLOR farbe1,0
332 u0 CLS
333 jL RETURN
334 uy menu29:
335 cr IF index=0 THEN CALL meldung("Buchungsspeicher leer"):RETU
    N
336 lc GOSUB schreiben
337 nP RETURN
338 oG menu31:
339 i7 CLS
340 lr CHDIR"df0:"
341 al COLOR farbe7,farbe0
342 Um FILES
343 MP CALL meldung ("Maustaste")
344 lQ COLOR farbe1,farbe0
345 7D CLS
346 Ju CHDIR"df0:Schublade"
347 xZ RETURN
348 OT menu32:
349 BH CLS
350 v1 CHDIR"df0:"
351 kv COLOR farbe7,farbe0
352 ew FILES
353 jt COLOR farbe6,farbe0
354 j3 INPUT "Bitte Filenamen eingeben ",a$
355 HN CLS
356 Xc COLOR farbe1,farbe0
357 oh KILL a$
358 V6 CHDIR"df0:Schublade"
359 91 RETURN
360 E1 menu33:
361 v0 CHDIR "df0:Schublade"
362 uF GOSUB einlesen
363 OW BEEP
364 Eq RETURN
365 lJ menu41:
366 y6 GOSUB systemzeit
367 FG GOTO einlesen
368 Qp menu42:
369 j1 WINDOW 5,,(0,100)-(maxbreite,147),16,1
370 iW LINE (80,5)-(130,50),farbe3,bf
371 ZE LINE (235,5)-(285,50),farbe3,bf
372 Rr LINE (390,5)-(440,50),farbe3,bf
373 DP LINE (540,17)-(610,37),farbe3,bf
374 JW COLOR farbe6,farbe3
375 Nt LOCATE 4,12
376 XN PRINT USING "###.###.###.###";jahr
377 k5 LOCATE 4,33:PRINT USING "###.###.###.###";monat
378 qq LOCATE 4,52:PRINT USING "###.###.###.###";tag
379 z0 LOCATE 4,70:PRINT "fertig";
380 v0 COLOR farbe1,farbe0
381 VT LOCATE 4,5:PRINT "Jahr";
382 qc LOCATE 4,23:PRINT "Monat";
383 Hv LOCATE 4,45:PRINT "Tag";
384 Tg COLOR farbe6,farbe3
385 wc menu421:
386 q1 WHILE MOUSE(0)<>0:WEND
387 Do WHILE MOUSE(0)=0:WEND
388 xY x=MOUSE(1):y=MOUSE(2)
389 yI IF x>540 AND x<610 AND y>17 AND y<37 THEN menu422
390 fD IF y<5 AND y>50 THEN menu421
391 Xh IF y<27 THEN
392 oB2 IF x>80 AND x<130 THEN jahr=jahr+1
393 9p IF x>235 AND x<285 THEN monat=monat+1:IF monat>12 THEN
    monat=1
394 eX IF x>390 AND x<440 THEN tag=tag+1:IF tag>maxtag THEN t
    ag=1
395 yH0 ELSE
396 OP2 IF x>80 AND x<130 THEN jahr=jahr+1
397 bJ IF x>235 AND x<285 THEN monat=monat+1:IF monat<1 THEN
    monat=12
398 xW IF x>390 AND x<440 THEN tag=tag+1:IF tag<1 THEN tag=ma
    xtag
399 NG0 END IF
400 Ma jahr$=RIGHT$(STR$(jahr),LEN(STR$(jahr))-1)
401 YY monat$=RIGHT$(STR$(monat),LEN(STR$(monat))-1):IF LEN(monat$

```

```

)=1 THEN monat$="0"+monat$
402 s0 tag$=RIGHT$(STR$(tag),LEN(STR$(tag))-1):IF LEN(tag$)=1 THEN
    tag$="0"+tag$
403 G8 IF jahr$+monat$<"198801" THEN jahr=1988:monat=1
404 Uw IF jahr$>RIGHT$(DATE$,4) THEN jahr=VAL(RIGHT$(DATE$,4)):mo
    nat=12
405 eE maxtag=28+(monat\7+2)*(1-monat MOD 2)+(3-monat\9)*(monat MO
    D 2)+(monat=2)*2+(jahr MOD 4=0)*(monat=2)
406 cJ IF tag>maxtag THEN tag=maxtag
407 qo LOCATE 4,12:PRINT USING "###.###.###.###";jahr;
408 rS LOCATE 4,33:PRINT USING "###.###.###.###";monat;
409 5W LOCATE 4,52:PRINT USING "###.###.###.###";tag;
410 La GOTO menu421
411 P6 menu422:
412 Ym jahr$=RIGHT$(STR$(jahr),LEN(STR$(jahr))-1)
413 kk monat$=RIGHT$(STR$(monat),LEN(STR$(monat))-1):IF LEN(monat$
    )=1 THEN monat$="0"+monat$
414 4C tag$=RIGHT$(STR$(tag),LEN(STR$(tag))-1):IF LEN(tag$)=1 THEN
    tag$="0"+tag$
415 8r WINDOW CLOSE 5
416 7T WINDOW OUTPUT 1
417 n8 GOSUB einlesen
418 61 RETURN
419 Jd menu51:
420 KQ CLS
421 9W FOR i=0 TO 7
422 ge LINE (i*70+10,20)-(i*70+50,50),1,b:REM Farbfelder
423 hy PAINT (i*70+15,30),i,1
424 q8 LOCATE 8,1
425 dK PRINT PTAB(i*70+20);i
426 JZ NEXT i
427 gQ LOCATE 2,2:PRINT "Farben"
428 ES LOCATE 12,2:PRINT "rot"
429 tI LOCATE 14,2:PRINT "grün"
430 Xs LOCATE 16,2:PRINT "blau"
431 4M FOR i=0 TO 2
432 Ya LINE (100,86+i*16)-(400,96+i*16),1,b:REM Einzelfarben
433 Qg NEXT i
434 mZ LINE (130,145)-(220,165),1,b:PAINT (180,155),farbe3,1
435 Ip farbe=0:COLOR farbe6,farbe3
436 BD LOCATE 20,20:PRINT "fertig";
437 Js xpos(1)=rot(farbe+1)*291+101
438 zt xpos(2)=gruen(farbe+1)*291+101
439 cS xpos(3)=blau(farbe+1)*291+101
440 Jd FOR i=1 TO 3
441 iN LOCATE 10+i*2,1
442 ju PRINT PTAB(xpos(i));"] ";
443 aq NEXT i
444 PJ schritt=.05
445 yZ menu511:
446 Cs LINE (farbe*70+8,18)-(farbe*70+52,52),1,b
447 Cs LINE (farbe*70+7,17)-(farbe*70+53,53),1,b
448 U3 xpos(1)=rot(farbe+1)*291+101
449 A4 xpos(2)=gruen(farbe+1)*291+101
450 nd xpos(3)=blau(farbe+1)*291+101
451 2K COLOR 1,0
452 Vp FOR i=1 TO 3
453 DZ LOCATE 10+i*2,1
454 v6 PRINT PTAB(xpos(i));"] ";
455 m2 NEXT i
456 yq WHILE MOUSE(0)<>0:WEND
457 Lw WHILE MOUSE(0)=0:WEND
458 oc maus=MOUSE(0):xmouse=MOUSE(1):ymouse=MOUSE(2)
459 Ub IF ymouse>145 AND ymouse<165 AND xmouse>130 AND xmouse<
    220 THEN menu512
460 am IF ymouse>20 AND ymouse<50 THEN
461 nAR FOR i=0 TO 7
462 XC IF xmouse>i*70+10 AND xmouse<i
    *70+50 THEN
463 BSU COLOR 0,0
464 n3 FOR j=1 TO 3
465 Ro LOCATE 10+j*2,1
466 AM PRINT PTAB(xpos(j));"] ";
467 OH NEXT j
468 T8 LINE (farbe*70+8,18)-(farbe*7
    0+52,52),0,b
469 T8 LINE (farbe*70+7,17)-(farbe*7
    0+53,53),0,b
470 kV farbe=i
471 XQR END IF

```

Listing 1. (Fortsetzung)

```

472 3J      NEXT 1
473 Qa      GOTO menu511
474 aTO END IF
475 t7 IF ymouse>86 AND ymouse<96 THEN
476 OTU      COLOR 0,0:LOCATE 12,1
477 1N      IF xmouse<100 OR xmouse<xp
           s(1) THEN
           rot(farbe+1)=rot(farbe+1)-
           schritt
478 dxX      END IF
479 fYV      IF xmouse>400 OR xmouse>xp
480 Dp      os(1) THEN
           rot(farbe+1)=rot(farbe+1)+
           schritt
           END IF
481 cuX
482 1bV
483 gxO PRINT PTAB(xpos(1));"] ";
484 kd END IF
485 Xg IF ymouse>102 AND ymouse<112 THEN
486 KpU      COLOR 0,0:LOCATE 14,1
487 yZ      IF xmouse<100 OR xmouse<xp
           s(2) THEN
           gruen(farbe+1)=gruen(farbe
           +1)-schritt
488 VhX      END IF
489 p1V      IF xmouse>400 OR xmouse>xp
490 T1      os(2) THEN
           gruen(farbe+1)=gruen(farbe
           +1)+schritt
           END IF
491 MgX
492 s1V
493 tBO PRINT PTAB(xpos(2));"] ";
494 un END IF
495 Zw IF ymouse>118 AND ymouse<128 THEN
496 eBU      COLOR 0,0:LOCATE 16,1
497 E1      IF xmouse<100 OR xmouse<xp
           s(3) THEN
           blau(farbe+1)=blau(farbe+1
           )-schritt
498 CeX      END IF
499 zsV      IF xmouse>400 OR xmouse>xp
500 JD      os(3) THEN
           blau(farbe+1)=blau(farbe+1
           )+schritt
           END IF
501 7XX
502 2vV
503 6PO PRINT PTAB(xpos(3));"] ";
504 4x END IF
505 Hw IF rot(farbe+1)>1 THEN rot(farbe+1)=1
506 8J IF rot(farbe+1)<0 THEN rot(farbe+1)=0
507 7q IF gruen(farbe+1)>1 THEN gruen(farbe+1)=1
508 tY IF gruen(farbe+1)<0 THEN gruen(farbe+1)=0
509 WY IF blau(farbe+1)>1 THEN blau(farbe+1)=1
510 IG IF blau(farbe+1)<0 THEN blau(farbe+1)=0
511 5b PALETTE farbe,rot(farbe+1),gruen(farbe+1),blau(farbe+1)
512 3D GOTO menu511
513 7j menu512:
514 WF CALL requester ("Farben abspeichern", "Ja", "Nein")
515 xm IF req=1 THEN
516 Nc OPEN "O", #1, "Farben"
517 nC FOR i=1 TO 8
518 kx PRINT #1, rot(i)
519 sJ PRINT #1, gruen(i)
520 QW PRINT #1, blau(i)
521 q6 NEXT i
522 eR CLOSE #1
523 NG END IF
524 O6 CLS
525 pR RETURN
526 4P menu52:
527 Ok GOTO farben
528 8U menu53:
529 fK IF index<>0 THEN
530 1D BEEP:BEEP:BEEP
531 kC CALL meldung("Buchungsspeicher entleeren !!!")
532 wY RETURN
533 Cv ELSE
534 oC MENU RESET
535 pV WINDOW CLOSE 2
536 1Q WINDOW CLOSE 1
537 RW SCREEN CLOSE 1
538 bL SYSTEM
539 dW END IF
540 Mc menu61:
541 3R sum#=0
542 NK IF konto$="" THEN CALL meldung("Kein Konto definiert"):RETU
RN
543 xv OPEN "I", #1, konto$+jahr$+monat$

```

```

544 Zr OPEN "lpt1:" FOR OUTPUT AS #3
545 Eo PRINT #3, "Einzelkonto: ";konto$;TAB(40);"Jahr ";jahr$;TAB(
55);"Monat ";monat$
546 Vg PRINT #3,
547 Jr PRINT #3, "Tag";TAB(10);"Bezeichnung";TAB(40);"Betrag"
548 fh PRINT #3, STRING$(52, "_")
549 II WHILE NOT EOF(1)
550 cO INPUT #1, a$, b$
551 9u INPUT #1, a$, b$, c$
552 LS PRINT #3, a$;TAB(10);b$;TAB(40);
553 fp PRINT #3, USING "#####.###";VAL(c$)
554 gE sum#=sum#+VAL(c$)
555 OC WEND
556 np PRINT #3, STRING$(52, "_")
557 Ph PRINT #3, TAB(29);
558 Jh PRINT #3, USING "      Saldo:#####.###";sum#
559 F2 CLOSE #1
560 OD CLOSE #3
561 P1 RETURN
562 k1 menu62:
563 Pn sum#=0
564 Gr IF maxkonto=0 THEN RETURN
565 md OPEN "R", #1, jahr$+monat$, 8
566 vD OPEN "lpt1:" FOR OUTPUT AS #3
567 c4 FIELD #1, 8 AS a$
568 Ra PRINT #3, "Monatskonto: ";TAB(40);"Jahr ";jahr$;TAB(55);"Mon
at ";monat$
569 s3 PRINT #3,
570 V1 PRINT #3, "Konto";TAB(30);"Betrag"
571 xy PRINT #3, STRING$(42, "_")
572 ux FOR i=1 TO maxkonto
573 hN PRINT #3, verzeichnis$(i);TAB(30);
574 81 LSET a$="":GET #1, 1
575 km PRINT #3, USING "#####.###";CVD(a$)
576 jJ sum#=sum#+CVD(a$)
577 kO NEXT i
578 45 PRINT #3, STRING$(42, "_")
579 5p PRINT #3, TAB(22);
580 r7 PRINT #3, USING "      Saldo:#####.###";sum#
581 4F PRINT #3,
582 cP CLOSE #1
583 1a CLOSE #3
584 mO RETURN
585 9R menu63:
586 mA sum#=0
587 dE IF maxkonto=0 THEN RETURN
588 xJ OPEN "R", #1, jahr$, 8
589 Ia OPEN "lpt1:" FOR OUTPUT AS #3
590 zR FIELD #1, 8 AS a$
591 Jc PRINT #3, "Jahreskonto: ";TAB(40);"Jahr ";jahr$;TAB(55);"Mona
t ";monat$
592 FQ PRINT #3,
593 s8 PRINT #3, "Konto";TAB(30);"Betrag"
594 KL PRINT #3, STRING$(42, "_")
595 HK FOR i=1 TO maxkonto
596 4k PRINT #3, verzeichnis$(i);TAB(30);
597 V8 LSET a$="":GET #1, 1
598 79 PRINT #3, USING "#####.###";CVD(a$)
599 6g sum#=sum#+CVD(a$)
600 7N NEXT i
601 RS PRINT #3, STRING$(42, "_")
602 SC PRINT #3, TAB(22);
603 ms PRINT #3, USING "      Saldo:#####.###";sum#
604 y1 CLOSE #1
605 7w CLOSE #3
606 8k RETURN
607 XJ menu71:
608 U8 drucker CHR$(27)+CHR$(64)+CHR$(27)+"1"+CHR$(10)+CHR$(27)+"N
"+CHR$(4)
609 Bn RETURN
610 cp menu72:
611 mU drucker CHR$(12)
612 Eq RETURN
613 hv menu73:
614 pY drucker CHR$(27)+"1"+CHR$(0)+CHR$(27)+"x1"
615 Ht RETURN
616 m1 menu74:
617 98 drucker CHR$(27)+"x0"+CHR$(27)+"!"+CHR$(0)
618 Kw RETURN
619 r7 menu75:
620 EE drucker CHR$(27)+"x0"+CHR$(27)+"!"+CHR$(1)
621 Nz RETURN
622 aW unterbrechung:
623 Fd MENU RESET
624 Ad CHDIR "df0:Schublade"

```



```

625 pk END
626 3y fehler:
627 Gm BEEP
628 k1 IF ERR=53 THEN CALL meldung("File existiert nicht"):RESUME
weiter
629 gz nummer$=STR$(ERR)
630 PL CALL meldung("Fehler "+nummer$)
631 N1 MENU RESET
632 I1 CHDIR "d0:Schublade"
633 xs END
634 yQ einlesen:
635 Fp RESTORE einlesen
636 QU DATA "Verzeichnis"
637 UQ DATA "Option", "Buchen", "neues Konto",
"Konto löschen"
638 Yz DATA "Übersicht Einzelkonto", "Übersicht Monat",
"Übersicht Jahr", "Buchungen", "Eintrag",
en, "##"
639 dn DATA "Disk", "Directory", "File löschen", "Diskwechsel", "##"
640 2I DATA "Zeit", "Systemzeit", "Bezugszeit", "##"
641 6I DATA "System", "Farben", "Farbreset", "Ende", "##"
642 Tk DATA "Ausgabe", "Einzelkonto", "Monatskonto", "Jahreskonto", "##"
643 ud DATA "Drucker", "RESET", "Vorschub", "NLQ", "DRAFT",
"ELITE", "$"
644 ay MENU RESET
645 vW konto$=""
646 ya READ verzeichnis$:MENU 1,0,1,verzeichnis$
647 kk OPEN "A", #1, verzeichnis$+jahr$:CLOSE #1
648 IL OPEN "I", #1, verzeichnis$+jahr$
649 2B maxkonto=0
650 vv WHILE NOT EOF(1)
651 hu maxkonto=maxkonto+1
652 Sf INPUT #1, verzeichnis$(maxkonto)
653 Ek INPUT #1, flag$(maxkonto)
654 UE MENU 1, maxkonto, flag$(maxkonto), verzeichnis$(maxkonto)
655 Oo WEND
656 ob CLOSE #1
657 Un i=2:j=0
658 9E einles:
659 Mz READ a$
660 aG IF a$="# " THEN i=i+1:j=0:GOTO einles
661 Wa IF a$<">" THEN MENU 1,j,1,a$:j=j+1:GOTO einles
662 2e RETURN
663 2R bildschirm:
664 QD SCREEN 1,640,256,3,2
665 fu maxbreite=631:maxhoehe=242
666 mp WINDOW 2,,(0,215)-(maxbreite,maxhoehe),16,1:COLOR 1,0
667 cF WINDOW 1,"von A.N. u. M.F."+SPACE$(10)+"++Haushaltsverwal
tung++"+SPACE$(20)+"c 1988", (0,0)-(maxbreite,200),16,1
668 8k RETURN
669 G3 farben:
670 AI RESTORE farben
671 Fb DATA 0,0,0
672 Sm DATA 1,1,1
673 In DATA .4,.6,1
674 XJ DATA 1,.73,0
675 XM DATA .33,.87,0
676 FB DATA .73,1,0
677 6Y DATA .93,.2,0
678 gO DATA 1,.6,.67
679 Jg FOR i=0 TO 7
680 5h READ rot(i+1),gruen(i+1),blau(i+1)
681 BN PALETTE 1,rot(i+1),gruen(i+1),blau(i+1)
682 Rh NEXT i
683 Nz RETURN
684 yy systemzeit:
685 gz jahr$=RIGHT$(DATE$,4):jahr=VAL(jahr$)
686 az monat$=LEFT$(DATE$,2):monat=VAL(monat$)
687 Xp tag$=MID$(DATE$,4,2):tag=VAL(tag$)
688 Tp datum$=tag$+"-"+monat$+"-"+jahr$
689 T5 RETURN
690 HB SUB meldung(a$) STATIC
691 98 SHARED maxbreite
692 43 a=LEN(a$)*8
693 Cy WINDOW 3,,(maxbreite-a-60,0)-(maxbreite,15),16,1
694 kq CLS
695 Da COLOR 6,0
696 bg LOCATE 2,(a+60)/8/2-LEN(a$)/2
697 M1 PRINT a$:COLOR 1,0
698 sk WHILE MOUSE(0)<>0:WEND
699 qv WHILE MOUSE(0)=0
700 JX WEND
701 14 WINDOW OUTPUT 1

```

```

702 bI WINDOW CLOSE 3
703 NP END SUB
704 qH SUB requester(a$,b$,c$) STATIC
705 Y1 SHARED req
706 6A a=LEN(a$)*8+60:b=LEN(b$)*8+60:c=LEN(c$)*8+60
707 m7 IF a>b+c THEN
708 bOD maxleng=a
709 219 ELSE
710 wJD maxleng=b+c
711 P10 END IF
712 8R WINDOW 5,,(0,0)-(maxleng,40),16,1
713 KG LOCATE 2,1:COLOR 6,0
714 ea PRINT PTAB((maxleng-LEN(a$)*8)/2);a$;
715 1o LINE (20,30)-(b-20,40),1,b:PAINT (22,35),2,1
716 v8 LINE (maxleng-c+20,30)-(maxleng-20,40),1,b:PAINT (maxleng-c
+22,35),2,1
717 Rk COLOR 0,2
718 Od LOCATE 5,1
719 CR PRINT PTAB(30);b$;
720 Qf LOCATE 5,1
721 Sy PRINT PTAB(maxleng+30-c);c$;
722 ZG reqes1:
723 H9 WHILE MOUSE(0)<>0:WEND
724 eF WHILE MOUSE(0)=0:WEND
725 Uf IF MOUSE(2)<30 OR MOUSE(2)>40 THEN reqes1
726 UN IF MOUSE(1)>20 AND MOUSE(1)<b-20 THEN req=1:GOTO reqes2
727 zs IF MOUSE(1)>maxleng-c+20 AND MOUSE(1)<maxleng-20 THEN req
=0:GOTO reqes2
728 fv GOTO reqes1
729 jR reqes2:
730 Dw WINDOW CLOSE 5
731 CY WINDOW OUTPUT 1
732 qs END SUB
733 R1 uhr:
734 OV MENU STOP
735 Ef fenster=WINDOW(1)
736 NF WINDOW OUTPUT 2
737 7d LOCATE 1,15
738 fN COLOR farbe5,0
739 mJ PRINT TIME$
740 Sm LOCATE 1,45
741 D5 COLOR farbe2,farbe0:zeit=INT(TIMER)
742 9z stunde=(zeit-arbeitsdauer)\3600
743 In minute=(zeit-arbeitsdauer-stunde*3600)\60
744 3E sekunde=INT(zeit-arbeitsdauer-stunde*3600-minute*60+.5)
745 Aa PRINT USING "h:##";stunde;
746 M2 PRINT USING "m:##";minute;
747 vg PRINT USING "s:##";sekunde;
748 i4 LOCATE 3,45
749 NF PRINT USING "## ## ##";FRE(-1);FRE(-2);FRE(0);
750 5x WINDOW OUTPUT fenster
751 L1 MENU ON
752 U6 RETURN
753 d2 aktuell:
754 fx WINDOW OUTPUT 2
755 we COLOR farbe5,0
756 U1 LOCATE 2,15
757 jg PRINT datum$
758 a8 LOCATE 3,15
759 Xk PRINT tag$+"-"+monat$+"-"+jahr$;
760 gF LOCATE 4,15
761 ks PRINT SPACE$(20);
762 iH LOCATE 4,15
763 I1 PRINT konto$;
764 uF LOCATE 2,45
765 PP PRINT USING "##";index;
766 17 WINDOW OUTPUT 1
767 jL RETURN
768 3R schreiben:
769 UO FOR zaehler=1 TO index
770 Qz OPEN "A", #1, konto$(zaehler)+jahr$(zaehler)+monat$(zaehler)
771 ZT PRINT #1, jahr$(zaehler)
772 I3 PRINT #1, monat$(zaehler)
773 H9 PRINT #1, tag$(zaehler)
774 QK PRINT #1, buchung$(zaehler)
775 Kx PRINT #1, betrag$(zaehler)
776 kX CLOSE #1
777 tW OPEN "R", #1, jahr$(zaehler),8
778 gL OPEN "R", #2, jahr$(zaehler)+monat$(zaehler),8
779 2U FIELD #1,8 AS a$
780 De FIELD #2,8 AS b$
781 48 GET #1,nummer(zaehler):a$=CVD(a$)
782 jX LSET a$=MKD$(VAL(betrag$(zaehler))+a$)

```

Listing 1. (Fortsetzung)

```

783 IG PUT #1,nummer(zaehler)
784 GN GET #2,nummer(zaehler):b# =CVD(b$)
785 g9 LSET b$=""
786 rh LSET b$=MKD$(VAL(betrag$(zaehler))+b#)
787 ON PUT #2,nummer(zaehler)
788 wj CLOSE #1
789 1p CLOSE #2
790 FX WINDOW OUTPUT 2
791 WE COLOR farbe5,0
792 Mh LOCATE 2,45
793 UH PRINT USING "##";index-zaehler;
794 DZ WINDOW OUTPUT 1
795 19 NEXT zaehler
796 9q index=0
797 PV CLS
798 Eq RETURN
799 rk SUB neudat(file$) STATIC
800 j6 OPEN "R",#1,file$,8
801 Oq FIELD #1,8 AS a$
802 OL LSET a$=MKD$(0)
803 gN FOR i=1 TO 19
804 ZU PUT #1,i
805 Qg NEXT i
806 E1 CLOSE #1
807 QB OPEN "O",#1,"zeit"

```

```

808 FH PRINT #1,RIGHT$(DATE$,4)+LEFT$(DATE$,2)
809 H4 CLOSE #1
810 68 END SUB
811 eQ pruefung:
812 dm OPEN "I",#1,"Zeit"
813 Ua INPUT #1,zeit$
814 M9 CLOSE #1
815 UB IF LEFT$(zeit$,4)<RIGHT$(DATE$,4) THEN
816 6v8 CALL neudat(RIGHT$(DATE$,4))
817 Y4 FOR j=1 TO 12
818 Xh j$=STR$(j):j$=RIGHT$(j$,LEN(j$)-1)
819 oZ IF LEN(j$)=1 THEN j$="0"+j$
820 ws CALL neudat(RIGHT$(DATE$,4)+j$)
821 iz NEXT j
822 DJ CALL meldung ("Jahresdatei geschrieben")
823 D60 END IF
824 eG RETURN
825 3T SUB drucker(a$) STATIC
826 O6 OPEN "par:" FOR OUTPUT AS #3
827 1F PRINT #3,a$;
828 1X CLOSE #3
829 PR END SUB
(C) 1987 M&T

```

Listing 1. (Schluß)

Fortsetzung von Seite 27

Intelligenter Sprachentrainer

```

13 pU6 READ t$(1)
14 pc4 IF t$(1)<>"-1" THEN NEXT
15 Ow FOR i=1 TO 10
16 c76 READ ff(i)
17 MR4 NEXT
18 pT FOR i=1 TO 16
19 K16 READ f!(i)
20 PU4 NEXT
21 kZ anwenderladen
22 Db0 REM Bildschirmaufbau
23 vk4 SCREEN 1,640,244,3,2
24 8C WINDOW 1,"Vokabel - Trainer Vers. 1.0
von Andreas Regul (c) 7/1988", (0,0)-(631,230),
0,1
25 f0 FOR i=1 TO 4
26 Oy6 MENU 1,0,0,""
27 Wb4 NEXT
28 oB FOR i=0 TO 7
29 gu6 FOR j=0 TO 2
30 Q08 READ farb(i,j)
31 af6 NEXT
32 gu PALETTE 1,f!(farb(i,0)),f!(farb(i,1)),f!(farb(i,2))
33 ch4 NEXT
34 Zv COLOR 3,2
35 gh e$=DATE$
36 TL dat$=MID$(e$,4,2)+"."+LEFT$(e$,2)+"."+RIGHT$(e$,4)
37 y3 fenster 0,0,632,23
38 3D funktion 1
39 5V pixlocate 348,16:PRINT "Datum: " dat$
40 Jq pixlocate 502,16:PRINT "Zeit: " TIME$
41 Px ON TIMER (1) GOSUB uhr
42 JP TIMER ON
43 A80 REM Anwendernamen eingeben
44 oP2 anweingabe:
45 az4 fenster 0,30,320,48
46 vV TIMER OFF
47 z3 pixlocate 16,49:PRINT "Bitte den Anwendernamen eingeb
en:"
48 PV TIMER ON
49 GQ kasten 16,60,181,11
50 81 z$=""
51 kb stringeingabe 21,68,20,5
52 Au z$=UCASE$(z$)
53 Mv FOR i=1 TO man
54 ga IF na$(i)<>z$ THEN NEXT:GOSUB neuanwender ELSE anw=
1:GOSUB anwbenutzung
55 Fc FOR i=0 TO 7
56 7L6 FOR j=0 TO 2

```

```

57 Bj8 farb(i,j)=fa(anw,i,j)
58 166 NEXT
59 274 NEXT
60 Kh FOR i=0 TO 7
61 9N6 PALETTE 1,f!(farb(i,0)),f!(farb(i,1)),f!(farb(i,2))
62 5A4 NEXT
63 n1 mabn=mabn+1
64 2A IF mabn>10 THEN mabn=10:FOR i=0 TO 9:FOR j=0 TO 2:ab
n$(i,j)=abn$(i+1,j):NEXT j,i
65 JG abn$(mabn,0)=dat$
66 yz abn$(mabn,1)=LEFT$(TIME$,5)
67 tC datnamenlad
68 It0 REM Hauptmenue
69 sd2 hauptmenu:
70 jt4 loeschen
71 ak funktion 1
72 X3 menue 0,30,210,2,10
73 YS IF be THEN hauptmenu
74 TO ON aw GOSUB diskette,eingabe,ausgabe,editieren,suche,
abfrage,statistik,ende,sonderfunktion
75 8Y GOTO hauptmenu
76 Us0 REM Diskettenbetrieb
77 Nz2 diskette:
78 nt4 funktion 2
79 fy menue 100,60,220,11,14
80 xK IF be THEN RETURN
81 JQ ON aw GOSUB laden,speichern,datloeschen,wechseln
82 gI RETURN
83 5D0 REM Vokabeldatei laden
84 IA2 laden:
85 y84 loeschen
86 No funktion 11
87 cT IF md=0 THEN fehler 0,30,31:RETURN
88 XU IF va THEN frage 0,30,35:IF aw=0 THEN RETURN
89 hx dateiauswahl
90 7U IF be THEN RETURN
91 C9 e$=d$(aw)+"."+LEFT$(na$(anw),3)
92 ux lad=aw
93 sS OPEN e$ FOR APPEND AS #1
94 7G e=LOF(1)
95 L5 CLOSE 1
96 g4 IF e=0 THEN KILL e$:fehler 50,58,34:RETURN
97 DD OPEN e$ FOR INPUT AS #1
98 6e INPUT #1,mbn
99 Ak FOR i=1 TO mbn
100 p36 FOR j=0 TO 2
101 HP8 INPUT #1,bn$(i,j)
102 Jo6 NEXT
103 kp4 NEXT
104 4g INPUT #1,mv
105 Gu FOR i=1 TO mv
106 s56 FOR j=0 TO 1
107 DV8 LINE INPUT #1,v$(i,j)
108 pu6 NEXT

```



```

109 70      FOR j=0 TO 5
110 CG8      INPUT #1,st(i,j)
111 sx6      NEXT
112 ty4      NEXT
113 dN      CLOSE 1
114 v5      va=0
115 Dp      RETURN
116 3W0      REM Vokabeldatei speichern
117 In2      speichern:
118 Vp4      loeschen
119 w0      funktion 12
120 we      IF mv=0 THEN fehler 0,30,32:RETURN
121 7S      fenster 0,30,344,57
122 9j      TIMER OFF
123 OM      COLOR 3,2
124 sY      pixlocate 16,48:PRINT "Dateiname      : "
125 yq      pixlocate 16,62:PRINT "Dateilänge      : " mv "Vokabeln"
126 3X      pixlocate 16,78:PRINT "Erste Eingabe:"
127 OJ      kasten 136,40,178,11
128 yD      pixlocate 143,78:IF lad THEN PRINT da$(lad) ELSE PRIN
T dat$
129 6S      COLOR 3,2
130 jP      TIMER ON
131 RB      z$=d$(lad)
132 uR      stringeingabe 140,48,20,5
133 eB      IF z$="" THEN RETURN
134 UE      z$=UCASE$(z$)
135 Ow      IF z$=d$(lad) THEN datspeichern
136 bx      FOR i=1 TO md
137 Tg6      IF UCASE$(d$(i))=z$ THEN fehler 100,60,36:RETURN
138 JO4      NEXT
139 FV      IF md>99 THEN fehler 100,80,38:RETURN
140 Aa      st=0
141 8t2      datspeichern:
142 nP4      e$=z$+" "+LEFT$(na$(anw),3)
143 6H      OPEN e$ FOR OUTPUT AS #1
144 Fk      PRINT #1,mbn
145 uU      FOR i=1 TO mbn
146 Zn6      FOR j=0 TO 2
147 QV8      PRINT #1,bn$(i,j)
148 TY6      NEXT
149 UZ4      NEXT
150 Dm      PRINT #1,mv
151 Oe      FOR i=1 TO mv
152 cp6      FOR j=0 TO 1
153 OX8      PRINT #1,v$(i,j)
154 Ze6      NEXT
155 r8      FOR j=0 TO 5
156 LM8      PRINT #1,st(i,j)
157 ch6      NEXT
158 di4      NEXT
159 N7      CLOSE 1
160 4k      IF st THEN RETURN
161 gq      va=0
162 PY      IF z$=d$(lad) THEN RETURN
163 Me      FOR i=md TO 0 STEP -1
164 OI      IF d$(i)>z$ THEN NEXT
165 Kv      FOR j=md+1 TO i+2 STEP -1
166 X36      d$(j)=d$(j-1)
167 j1      da$(j)=da$(j-1)
168 ns4      NEXT
169 6W      d$(i+1)=z$:da$(i+1)=dat$
170 Ig      md=md+1:lad=i+1
171 Ok      datnamensp
172 8k      RETURN
173 t90      REM Vokabeldatei loeschen
174 Ai2      datloeschen:
175 Qa4      loeschen
176 tM      funktion 13
177 4v      IF md=0 THEN fehler 0,30,31:RETURN
178 80      dateiauswahl
179 1C      n=aw
180 81      IF n=lad THEN lad=0
181 DJ      IF lad>n THEN lad=lad-1
182 by      IF be THEN RETURN
183 Fn      frage 40,80,37
184 mT      IF aw=0 THEN RETURN
185 Rb      e$=d$(n)+" "+LEFT$(na$(anw),3)
186 B8      KILL e$
187 N1      FOR i=n TO md
188 eB6      d$(i)=d$(i+1)
189 o7      da$(i)=da$(i+1)

```

```

190 9E4      NEXT
191 lu      md=md-1
192 j5      datnamensp
193 T5      RETURN
194 ZU0      REM Diskette wechseln
195 Uu2      wechseln:
196 lv4      loeschen
197 Gk      funktion 14
198 Ec      IF va AND lad THEN frage 0,30,39:IF aw=0 THEN RETURN
199 oy      loeschen
200 RC      fehler 0,30,40
201 kc      CHDIR "DFO:"
202 4N      datnamenlad
203 q2      IF be THEN fehler 0,30,41:RETURN
204 zL      fenster 0,30,632,190
205 U4      TIMER OFF
206 c4      pixlocate 16,47:PRINT "Vokabeldateien"
207 3D      pixlocate 250,47:PRINT "Erstellungsdatum"
208 z5      TIMER ON
209 cB      y=54
210 n9      FOR i=1 TO md
211 T96      y=y+10
212 om      IF y>210 THEN y=204:SCROLL (8,55)-(340,204),0,-10
213 cC      TIMER OFF
214 Iq      pixlocate 16,y:PRINT d$(i)
215 kZ      pixlocate 250,y:PRINT da$(i)
216 7D      TIMER ON
217 af4      NEXT
218 O3      WHILE MOUSE(0)<0:WEND
219 V6      WHILE MOUSE(0)=0:WEND
220 uW      RETURN
221 BPO      REM Vokabeln eingeben
222 Xd2      eingabe:
223 OZ4      WHILE INKEY$<>"":WEND
224 DN      loeschen
225 GI      funktion 3
226 rB      vokfenster
227 fD      franzfont 0,116
228 h5      fenster 260,116,372,83
229 sS      TIMER OFF
230 j5      COLOR 3,2
231 TI      pixlocate 272,133:PRINT "Information:"
232 bq      pixlocate 272,149:PRINT "Vokabelanzahl : "
233 Lz      pixlocate 272,163:PRINT "Speicherbedarf:"
234 7h      kasten 400,141,51,11
235 WB      kasten 400,155,51,11
236 hO      kasten 560,141,51,11
237 5U      COLOR 3,5
238 9H      pixlocate 406,149:PRINT USING "### ";mv
239 o2      pixlocate 406,163:PRINT USING "### ";mv*100/mmV
240 tF      COLOR 3,2
241 19      pixlocate 462,149:PRINT "von maximal"
242 AZ      COLOR 3,5
243 Nt      pixlocate 566,149:PRINT USING "### ";mmv
244 Zf      TIMER ON
245 IJ      kasten 272,173,339,11
246 SK      LINE (272,175)-(272+mv*336/mmV,181),3,bf
247 2n2      vokeingabe:
248 ET4      mv=mv+1
249 19      IF mv>mmv THEN mv=mmv:fehler 100,100,33:RETURN
250 Dn      TIMER OFF
251 J1      COLOR 3,5
252 oe      pixlocate 16,61:PRINT SPACES$(LEN(v$(mv-1,0)))
253 tH      pixlocate 16,99:PRINT SPACES$(LEN(v$(mv-1,1)))
254 jP      TIMER ON
255 RK      z$=""
256 rs      stringeingabe 16,61,73,5
257 8a      IF z$="" THEN mv=mv-1:RETURN
258 7C      v$(mv,0)=z$:z$="":va=1
259 kw      stringeingabe 16,99,73,5
260 Bd      IF z$="" THEN mv=mv-1:RETURN
261 Co      v$(mv,1)=z$
262 Pz      TIMER OFF
263 Vu      COLOR 3,5

```

Listing 2. Das Hauptprogramm »Trainer HP« muß sich auf der Diskette im Laufwerk befinden, das im Ladeprogramm angegeben ist. Ansonsten erscheint »File not found«. Bitte mit dem Checksummer (Seite 159) eingeben.

```

264 Zh      pixlocate 406,149:PRINT USING "###";mv
265 V3      pixlocate 406,163:PRINT USING "###";mv*100/mmv
266 qU      LINE (272+(mv-1)*336/mmv,175)-(272+mv*336/mmv,181),3,
           bf
267 w2      TIMER ON
268 DX      GOTO vokeingabe
269 Cx0     REM Vokabeln ausgeben
270 502     ausgabe:
271 y84     loeschen
272 75      funktion 4
273 RA      IF mv=0 THEN fehler 0,30,42:RETURN
274 Wj      fenster 0,30,350,75
275 FN      fenster 24,116,350,75
276 YV      information 380,30
277 8R      fenster 396,116,236,75
278 Sq      kasten 408,126,205,11
279 Hd      kasten 408,142,205,11
280 qJ      kasten 408,158,205,11
281 f6      kasten 408,174,205,11
282 Jj      TIMER OFF
283 pE      COLOR 3,5
284 h1      pixlocate 418,134:PRINT "1 10 Vokabel zurueck"
285 Wq      pixlocate 418,150:PRINT "1 10 Vokabel weiter"
286 Ea      pixlocate 418,166:PRINT "Statistik ausgeben"
287 1A      pixlocate 418,182:PRINT "Zum Hauptmenü"
288 HN      TIMER ON
289 Fo      LINE (438,127)-(439,135),3,b
290 S5      LINE (476,127)-(477,135),3,b
291 xS      LINE (438,143)-(439,151),3,b
292 AJ      LINE (476,143)-(477,151),3,b
293 N1      n=1:n1=n
294 ON      z$=v$(n,0)
295 DJ      vokausage 11,46,z$
296 7V      z$=v$(n,1)
297 n6      vokausage 35,132,z$
298 j92     abfrage7:
299 hM4     WHILE MOUSE(0)<0:WEND
300 oP      WHILE MOUSE(0)=0:WEND
301 uP      x=MOUSE(3):y=MOUSE(4)
302 cH      IF y<120 THEN ausgrafik
303 dV      IF x<399 OR x>626 OR y>191 THEN abfrage7
304 Vk      IF 'y>172 THEN RETURN
305 WY      IF y>156 THEN vokstatist
306 VH      IF y>140 THEN IF x>439 AND x<480 THEN n=n+10:GOTO
           vokausage ELSE n=n+1:GOTO vokausage
307 MB      IF y>119 THEN IF x>439 AND x<480 THEN n=n-10 ELSE
           n=n-1
308 f02     vokausage:
309 JN4     IF n<1 THEN n=mv ELSE IF n>mv THEN n=1
310 GJ      LINE (11,38)-(338,102),2,bf
311 AZ      LINE (35,124)-(360,189),2,bf
312 If      z$=v$(n,0)
313 Vb      vokausage 11,46,z$
314 Pn      z$=v$(n,1)
315 50      vokausage 35,132,z$
316 Hr      TIMER OFF
317 Nm      COLOR 3,5
318 F7      pixlocate 572,78:PRINT USING "###";n
319 ms      TIMER ON
320 GT      LINE (n1*216/mmv+393,90)-(n1*216/mmv+393,96),3
321 c6      LINE (n*216/mmv+393,90)-(n*216/mmv+393,96),4
322 hx      n1=n
323 ih      GOTO abfrage7
324 D12     ausgrafik:
325 qo4     IF x<396 OR x>611 OR y<89 OR y>99 THEN abfrage7
326 qr      x=x-394
327 4C      n=CINT(x*mmv/216)
328 jR      IF n>mv THEN n=n1:GOTO abfrage7
329 Ou      GOTO vokausage
330 7p2     vokstatist:
331 TL4     fenster 0,195,632,35
332 X7      TIMER OFF
333 Ok      COLOR 3,2
334 XP      pixlocate 8,210:PRINT "Übersetzung abfragen:"
335 xJ      pixlocate 8,224:PRINT "Fremdwort abfragen :";
336 a1      pixlocate 346,210:PRINT "Schrift üben:";
337 gb      kasten 180,202,161,11
338 55      kasten 180,216,161,11
339 Fm      kasten 456,202,161,11
340 k9      COLOR 3,5
341 UX      pixlocate 184,210:PRINT USING "### von ";st(n,1);
342 z9      PRINT USING "### = ";st(n,0);

```

```

343 OS      IF st(n,0) THEN PRINT USING "### %";st(n,1)*100/st
           (n,0) ELSE PRINT " 0 %"
344 r1      pixlocate 184,224:PRINT USING "### von ";st(n,3);
345 8K      PRINT USING "### = ";st(n,2);
346 r0      IF st(n,2) THEN PRINT USING "### %";st(n,3)*100/st
           (n,2); ELSE PRINT " 0 %";
347 Wa      pixlocate 460,210:PRINT USING "### von ";st(n,5);
348 HV      PRINT USING "### = ";st(n,4);
349 Ka      IF st(n,4) THEN PRINT USING "### %";st(n,5)*100/st
           (n,4) ELSE PRINT " 0 %"
350 MO      LINE (580,216)-(616,226),5,bf
351 9o      rahmen 580,216,37,11
352 6C      pixlocate 582,224:PRINT " OK";
353 KQ      TIMER ON
354 Vt2     abfrage5:
355 bC4     WHILE MOUSE(0)<0:WEND
356 iJ      WHILE MOUSE(0)=0:WEND
357 oJ      x=MOUSE(3):y=MOUSE(4)
358 UZ      IF x<580 OR x>616 OR y<216 OR y>226 THEN abfrage5
359 kE      LINE (0,195)-(631,230),1,bf
360 JI      GOTO abfrage7
361 Vy0     REM Vokabeln editieren
362 MW2     editieren:
363 ga4     funktion 5
364 Un      IF mv=0 THEN fehler 100,70,43:RETURN
365 kN      menue 100,70,200,44,46
366 Zw      IF be THEN RETURN
367 ue      ON aw GOTO verbessern,vokloeschen,sploeschen
368 Bn0     REM Vokabeln verbessern
369 VR2     verbessern:
370 Zj4     loeschen
371 Me      funktion 44
372 9d      vokausage 11,46,z$
373 g3      IF be THEN RETURN
374 dn      loeschen
375 Ga      vokfenster
376 ZW      franzfont 0,120
377 Gq      TIMER OFF
378 M1      COLOR 3,5
379 ac      pixlocate 16,99:PRINT v$(aw,1)
380 lr      TIMER ON
381 Y7      WHILE INKEY$<>"":WEND
382 HH      z$=v$(aw,0)
383 uv      stringeingabe 16,61,73,5
384 pi      IF z$<>" " THEN v$(aw,0)=z$:va=1
385 QM      z$=v$(aw,1)
386 nz      stringeingabe 16,99,73,5
387 WA      IF z$<>" " THEN v$(aw,1)=z$
388 cE      RETURN
389 W10     REM Vokabeln loeschen
390 KF2     vokloeschen:
391 u44     loeschen
392 j2      funktion 45
393 Uy      vokausage 11,46,z$
394 10      IF be THEN RETURN
395 Vg      n=aw
396 hL      frage 50,70,47
397 Du      IF aw=0 THEN RETURN
398 wZ      FOR i=n TO mv
399 bo6     FOR j=0 TO 1
400 r68     v$(i,j)=v$(i+1,j)
401 Yd6     NEXT
402 q7      FOR j=0 TO 5
403 BS8     st(i,j)=st(i+1,j)
404 bg6     NEXT
405 ch4     NEXT
406 q7      mv=mv-1
407 ju      va=1
408 wY      RETURN
409 Ab0     REM Speicher loeschen
410 d02     sploeschen:
411 EO4     loeschen
412 5P      funktion 46
413 56      IF va THEN frage 0,30,48:IF aw=0 THEN RETURN
414 vC      IF va=0 THEN frage 0,30,49:IF aw=0 THEN RETURN
415 Gu      FOR i=1 TO mv
416 AL6     FOR j=0 TO 5
417 lJ8     st(i,j)=0
418 pu6     NEXT
419 qv4     NEXT
420 OL      mv=0:va=0:lad=0:mbn=0
421 91      RETURN

```



```

422 r40 REM Vokabeln suchen
423 GB2 suche:
424 lb4 funktion 6
425 JM menue 100,84,200,15,16
426 Xu IF be THEN RETURN
427 vd ON aw GOTO spsuche,disksuche
428 la0 REM Vokabeln im Speicher suchen
429 VD2 spsuche:
430 Xh4 loeschen
431 vd IF mv=0 THEN fehler 0,30,50:RETURN
432 bo GOSUB suchbegriff
433 U1 IF z$="" THEN RETURN
434 VN su$=UCASE$(z$)
435 cm loeschen
436 8L fenster 0,30,350,75
437 rz fenster 24,116,350,75
438 A7 information 380,30
439 eI FOR i=1 TO mv
440 Hr6 TIMER OFF
441 Nm COLOR 3,5
442 On pixlocate 572,78:PRINT USING "###";i
443 ms TIMER ON
444 97 LINE ((i-1)*216/mm+393,90)-((i-1)*216/mm+393,96),
3
445 DX LINE (i*216/mm+393,90)-(i*216/mm+393,96),4
446 nE IF INSTR(UCASE$(v$(i,0)),su$)=0 AND INSTR(UCASE$(v$
(i,1)),su$)=0 THEN nextvok
447 Tw LINE (11,38)-(338,102),2,bf
448 Nm LINE (35,124)-(360,189),2,bf
449 GY z$=v$(i,0)
450 io vokaussgabe 11,46,z$
451 Ng z$=v$(i,1)
452 Ib vokaussgabe 35,132,z$
453 Om frage 390,115,51
454 Tp IF aw THEN vokgef
455 wI LINE (390,110)-(630,170),1,bf
456 qL2 nextvok:
457 SX4 NEXT
458 ah fehler 390,115,52
459 lN RETURN
460 mt2 vokgef:
461 204 LINE (390,110)-(630,170),1,bf
462 hp fehler 390,115,53
463 pR RETURN
464 vP2 suchbegriff:
465 g54 fenster 0,30,632,47
466 hH TIMER OFF
467 Yu COLOR 3,2
468 qM pixlocate 12,46:PRINT "Bitte einen Suchbegriff eingeb
en:"
469 CI TIMER ON
470 pu LINE (12,56)-(611,66),5,bf
471 Y8 rahmen 12,56,600,11
472 wp z$=""
473 bf stringeingabe 16,64,73,5
474 Oc RETURN
475 NBO REM Vokabel auf Diskette suchen
476 Y42 disksuche:
477 IS4 loeschen
478 vm IF md=0 THEN fehler 0,30,31:RETURN
479 Qm fenster 0,30,632,190
480 vV TIMER OFF
481 m8 COLOR 3,2
482 vF pixlocate 12,47:PRINT "Bitte Dateien markieren, die d
urchsucht werden sollen:"
483 CY FOR i=1 TO md
484 iV6 dm(i)=0
485 uz4 NEXT
486 5e y=54
487 Gc FOR i=1 TO md
488 wc6 y=y+10
489 Tw pixlocate 20,y:PRINT d$(i)
490 pJ4 IF y<176 THEN NEXT
491 Ln kasten 16,198,121,15
492 U7 kasten 147,198,121,15
493 DN kasten 280,198,121,15
494 NR kasten 432,198,180,15
495 Fe COLOR 3,5
496 PA pixlocate 28,208:PRINT "letzte Datei"
497 ry pixlocate 156,208:PRINT "naechste Datei"
498 lK pixlocate 284,208:PRINT "Suche beginnen"
499 8G pixlocate 442,208:PRINT "Zurueck zum Hauptmenue"

```

```

500 5R COLOR 3,2
501 io TIMER ON
502 2V n=1:e=1:be=0
503 ze WHILE MOUSE(0)<0:WEND
504 Bd2 abfrage9:
505 714 WHILE MOUSE(0)=0:WEND
506 D1 x=MOUSE(3):y=MOUSE(4)
507 iF IF y<54 OR y>212 THEN abfrage9
508 5v IF y<198 THEN aw=INT((y-47)/10):GOTO datmark
509 mz IF x>431 THEN be=1:RETURN
510 fY IF x>279 THEN voksuchen
511 w3 IF md<14 THEN abfrage9
512 vE IF x>146 THEN n=n+1 ELSE IF x>15 THEN n=n-1
513 PY IF n=0 THEN n=1:GOTO abfrage9 ELSE IF n+12>md THEN n
=md-12:GOTO abfrage9
514 T3 TIMER OFF
515 Kg COLOR 3,2
516 kV IF n<e THEN datzurueck
517 la SCROLL (12,56)-(190,185),0,-10
518 iI IF dm(n+12) THEN COLOR 2,3 ELSE COLOR 3,2
519 BN pixlocate 12,184:PRINT " " d$(n+12) SPACE$(21-LEN(d$(
n+12)))
520 17 TIMER ON
521 4u e=n
522 34 GOTO abfrage9
523 OK2 datzurueck:
524 VP4 SCROLL (12,56)-(190,185),0,10
525 3U IF dm(n) THEN COLOR 2,3 ELSE COLOR 3,2
526 6g pixlocate 12,64:PRINT " " d$(n) SPACE$(21-LEN(d$(n)))
527 8E TIMER ON
528 B1 e=n
529 AB GOTO abfrage9
530 lG2 datmark:
531 9A4 IF aw+n-1>md THEN abfrage9
532 hN dm(aw+n-1)=1-dm(aw+n-1)
533 mM TIMER OFF
534 r7 IF dm(aw+n-1) THEN COLOR 2,3 ELSE COLOR 3,2
535 aP pixlocate 12,54+aw*10:PRINT " " d$(aw+n-1) SPACE$(21-
LEN(d$(aw+n-1)))
536 f1 COLOR 3,2
537 iO TIMER ON
538 YD WHILE MOUSE(0)<0:WEND
539 KL GOTO abfrage9
540 yd2 voksuchen:
541 KU4 loeschen
542 Na GOSUB suchbegriff
543 Gn IF z$="" THEN RETURN
544 H9 su$=UCASE$(z$)
545 OY loeschen
546 u7 fenster 0,30,350,75
547 dL fenster 24,116,350,75
548 wt information 380,30
549 2c TIMER OFF
550 8X COLOR 3,5
551 97 pixlocate 572,64:PRINT " 0"
552 OM pixlocate 572,78:PRINT " 0"
553 Ye TIMER ON
554 cG LINE (394,90)-(609,96),5,bf
555 M1 FOR i=1 TO md
556 4L6 IF dm(i)=0 THEN nextdat
557 Rv OPEN d$(i)+". "+LEFT$(na$(anw),3) FOR INPUT AS #1
558 qa INPUT #1,e
559 ko FOR j=1 TO e
560 Rd8 FOR k=1 TO 3
561 lPA INPUT #1,e$
562 9E8 NEXT
563 AF6 NEXT
564 wg INPUT #1,e
565 Is TIMER OFF
566 On COLOR 3,5
567 bK pixlocate 572,64:PRINT USING "###";e
568 nt TIMER ON
569 rV LINE (394,90)-(609,96),5,bf
570 8Y LINE (393,90)-(e*216/mm+393,96),3,bf
571 wO FOR j=1 TO e
572 Pz8 TIMER OFF
573 Vu COLOR 3,5
574 Bz pixlocate 572,78:PRINT USING "###";j
575 uO TIMER ON

```

Listing 2. (Fortsetzung)

```

576 m6 s=(j-1)*216/mmv+393:t=j*216/mmv+393
577 pU LINE (s,90)-(s,96),3
578 4h LINE (t,90)-(t,96),4
579 fB LINE INPUT #1,e$
580 2e LINE INPUT #1,s$
581 p2 FOR k=0 TO 5
582 aQA INPUT #1,s
583 UZ8 NEXT
584 ch IF INSTR(UCASE$(e$),su$)=0 AND INSTR(UCASE$(s$),s
u$)=0 THEN nextvok2
585 hA LINE (11,38)-(338,102),2,bf
586 b0 LINE (35,124)-(360,189),2,bf
587 OE z$=e$
588 w2 vokausage 11,46,z$
589 Yc z$=s$
590 Wp vokausage 35,132,z$
591 o0 frage 390,115,51
592 oB IF aw THEN CLOSE 1:GOTO vokgef
593 AW LINE (390,110)-(630,170),1,bf
594 eR2 nextvok2:
595 gl6 NEXT
596 QA CLOSE 1
597 Mw2 nextdat:
598 jo4 NEXT
599 ry fehler 390,115,52
600 2e RETURN
601 BD0 REM Vokabeln abfragen
602 jm2 abfrage:
603 2F4 IF mv=0 THEN CALL loeschen:fehler 0,30,54:RETURN
604 Op menu 100,70,210,17,20
605 Qn IF be THEN RETURN
606 JA IF aw=1 THEN einpraegen
607 H7 mbn=mbn+1
608 Je IF mbn>10 THEN mbn=10:FOR i=0 TO 9:FOR j=0 TO 2:bn$(
i,j)=bn$(i+1,j):NEXT j,i
609 Vx bn$(mbn,0)=dat$
610 p1 bn$(mbn,1)=LEFT$(TIME$,5)
611 fR ON aw-1 GOTO uebersetz,fremdwort,schrift
612 QK0 REM Vokabeln einpraegen
613 202 einpraegen:
614 Vf4 loeschen
615 6d funktion 17
616 2F fenster 0,30,350,75
617 lt fenster 24,116,350,75
618 41 information 380,30
619 FA fenster 400,116,232,67
620 Ex kasten 412,127,200,11
621 jk kasten 412,143,200,11
622 M7 kasten 412,165,200,11
623 Eo TIMER OFF
624 KJ COLOR 3,5
625 Dz pixlocate 420,135:PRINT "letzte Vokabel"
626 vF pixlocate 420,151:PRINT "nächste Vokabel"
627 r1 pixlocate 420,173:PRINT "Zurück zum Hauptmenü"
628 9V COLOR 3,2
629 ms TIMER ON
630 Tr z$=v$(1,0)
631 dJ vokausage 11,46,z$
632 az z$=v$(1,1)
633 DW vokausage 35,132,z$
634 mU n=1:e=1
635 Uy2 abfrage10:
636 8n4 WHILE MOUSE(0)<0:WEND
637 Fq WHILE MOUSE(0)=0:WEND
638 Lq x=MOUSE(3):y=MOUSE(4)
639 Fy IF x<413 OR x>611 OR y<125 OR y>177 THEN abfrage1
0
640 OG IF y>164 THEN RETURN
641 n1 IF y>140 THEN n=n+1 ELSE n=n-1
642 gk IF n<1 THEN n=mv ELSE IF n>mv THEN n=1
643 Y8 TIMER OFF
644 e3 COLOR 3,5
645 W0 pixlocate 572,78:PRINT USING "###";n
646 39 TIMER ON
647 aY s=e*216/mmv+393:t=n*216/mmv+393
648 7x e=n
649 ze LINE (s,90)-(s,96),3
650 Er LINE (t,90)-(t,96),4
651 lE LINE (11,38)-(338,102),2,bf
652 f4 LINE (35,124)-(360,189),2,bf
653 nA z$=v$(n,0)
654 06 vokausage 11,46,z$

```

```

655 uI z$=v$(n,1)
656 at vokausage 35,132,z$
657 09 GOTO abfrage10
658 Zy0 REM Uebersetzung abfragen
659 gd2 uebersetz:
660 rP4 funktion 18
661 C3 ab=1
662 jB GOTO abfragen
663 NT2 fremdwort:
664 xW4 funktion 19
665 B1 ab=0
666 DW2 abfragen:
667 jL4 menu 140,93,214,21,23
668 Ro IF be THEN RETURN
669 Je re=aw
670 41 ausmaske
671 x6 fenster 390,116,241,75
672 xf kasten 402,126,210,11
673 cG kasten 402,140,210,11
674 za kasten 402,156,72,11
675 CO kasten 540,156,72,11
676 Bw kasten 402,174,210,11
677 6g TIMER OFF
678 Cb COLOR 3,5
679 nh pixlocate 410,134:PRINT "insgesamt abgefragt: 0"
680 GO pixlocate 410,148:PRINT "davon gewußt : 0"
681 TT pixlocate 410,164:PRINT "richtig"
682 OR pixlocate 552,164:PRINT "falsch"
683 hq pixlocate 410,182:PRINT "Zurück zum Hauptmenü"
684 3P COLOR 3,2
685 gm TIMER ON
686 ne e=1
687 Lm2 bestfolge:
688 324 n=0
689 KV ON re GOSUB reihenfolge,gemischt,koennen
690 NA2 naechstvok:
691 dh4 n=n+1
692 G9 IF n>mv THEN bestfolge
693 Ru LINE (11,38)-(338,102),2,bf
694 Lk LINE (35,124)-(360,189),2,bf
695 Fr z$=v$(re(n),1-ab)
696 gm vokausage 11,46,z$
697 32 IF ab THEN re1=0 ELSE re1=2
698 R1 TIMER OFF
699 Xw COLOR 3,5
700 PH pixlocate 572,78:PRINT USING "###";n
701 SQ s=e*216/mmv+393:t=n*216/mmv+393
702 qV LINE (s,90)-(s,96),3
703 51 LINE (t,90)-(t,96),4
704 lr e=n
705 KT pixlocate 570,134:PRINT USING "###";st(re(n),re1
)
706 K6 pixlocate 570,148:PRINT USING "###";st(re(n),re1
+1)
707 Qm COLOR 3,2
708 39 TIMER ON
709 Jy WHILE MOUSE(0)<0:WEND
710 Q1 WHILE MOUSE(0)=0:WEND
711 W1 x=MOUSE(3):y=MOUSE(4)
712 51 IF x>400 THEN IF y>170 THEN abfrbeenden
713 vZ z$=v$(re(n),ab)
714 Wp vokausage 35,132,z$
715 rM2 abfrage11:
716 Q54 WHILE MOUSE(0)<0:WEND
717 X8 WHILE MOUSE(0)=0:WEND
718 d8 x=MOUSE(3):y=MOUSE(4)
719 GA IF x<402 OR x>611 OR y<154 OR y>190 THEN abfrage1
1
720 E1 IF y>170 THEN abfrbeenden
721 H6 IF x<506 THEN st(re(n),re1+1)=st(re(n),re1+1)+1
722 W2 st(re(n),re1)=st(re(n),re1)+1
723 k6 GOTO naechstvok
724 182 abfrbeenden:
725 ly4 bn$(mbn,2)=LEFT$(TIME$,5)
726 d9 IF lad THEN st=1:z$=d$(lad):GOTO datspeichern
727 5h RETURN
728 cG0 REM Schrift ueben
729 062 schrift:
730 pB4 funktion 20
731 lN menu 140,93,214,21,23
732 Tq IF be THEN RETURN
733 Lg re=aw

```



```

734 6k      ausgmaske
735 yX      franzfont 402,116
736 kp      fenster 0,196,350,35
737 4e      TIMER OFF
738 vH      COLOR 3,2
739 HK      pixlocate 12,211:PRINT "insgesamt abgefragt:"
740 f6      pixlocate 12,225:PRINT "davon gewußt      ":";
741 eH      kasten 180,203,50,11
742 3l      kasten 180,217,50,11
743 Fe      COLOR 3,5
744 fJ      kasten 260,203,70,11
745 4n      kasten 260,217,70,11
746 Oz      pixlocate 268,211:PRINT "richtig"
747 Xy      pixlocate 268,225:PRINT "falsch";
748 hn      TIMER ON
749 wz      ab=2:e=1
750 Qn2     bestfolge2:
751 434      n=0
752 LW      ON re GOSUB reihenfolge,gemischt,koennen
753 OT2     vokeingeben:
754 e14      n=n+1
755 Rc      IF n>mv THEN bestfolge2
756 Sv      LINE (11,38)-(338,102),2,bf
757 Ml      LINE (35,124)-(360,189),2,bf
758 5c      z$=v$(re(n),1)
759 hn      vokausgabe 11,46,z$
760 Rl      TIMER OFF
761 Xw      COLOR 3,5
762 PH      pixlocate 572,78:PRINT USING "####";n
763 SQ      s=e*216/mmv+393:t=n*216/mmv+393
764 qV      LINE (s,90)-(s,96),3
765 5i      LINE (t,90)-(t,96),4
766 1r      e=n
767 Xx      pixlocate 188,211:PRINT USING "####";st(re(n),4)
768 Ox      pixlocate 188,225:PRINT USING "####";st(re(n),5);
769 Qm      COLOR 3,2
770 39      TIMER ON
771 qP      WHILE INKEY$<>"":WEND
772 mf      z$=""
773 km      stringeingabe 35,132,40,2
774 oY      z$=UCASE$(z$)
775 9G      IF z$="" THEN abfrbeenden
776 JD      st(re(n),4)=st(re(n),4)+1
777 8J      e$=UCASE$(v$(re(n),0))
778 bG      IF e$=z$ THEN richtig
779 uX2     teilwort:
780 oE4      s=INSTR(e$,"")
781 Bu      IF s=0 THEN s=LEN(e$)+1
782 kq      IF LEFT$(e$,s-1)=z$ THEN richtig
783 hS      e$=MID$(e$,s+1)
784 MW      IF e$<>" " THEN teilwort
785 mE      LINE (262,204)-(327,212),5,bf
786 Lv      LINE (262,218)-(327,226),3,bf
787 sS      TIMER OFF
788 yN      COLOR 3,5
789 hg      pixlocate 268,211:PRINT "richtig"
790 wL      COLOR 5,3
791 Fg      pixlocate 268,225:PRINT "falsch";
792 vK      LINE (35,124)-(360,189),2,bf
793 oA      COLOR 3,2
794 b7      z$=v$(re(n),0)
795 p8      vokausgabe 35,132,z$
796 T2      TIMER ON
797 uN      FOR s=1 TO 3000:NEXT
798 kP      WHILE MOUSE(0)<0:WEND
799 7X      WHILE MOUSE(0)=0 AND INKEY$="" :WEND
800 ZD      GOTO vokeingeben
801 pt2     richtig:
802 zP4      LINE (262,204)-(327,212),3,bf
803 gI      LINE (262,218)-(327,226),5,bf
804 9J      TIMER OFF
805 Ba      COLOR 5,3
806 yx      pixlocate 268,211:PRINT "richtig"
807 Hg      COLOR 3,5
808 Wx      pixlocate 268,225:PRINT "falsch";
809 4Q      COLOR 3,2
810 hn      TIMER ON
811 1s      st(re(n),5)=st(re(n),5)+1
812 1P      GOTO vokeingeben
813 860     REM Statistik ausgeben
814 v12     statistik:
815 Gy4     funktion 8

```

```

816 VJ      IF mv=0 THEN CALL loeschen:fehler 0,30,55:RETURN
817 Of      menuue 100,90,250,24,26
818 rE      IF be THEN RETURN
819 68      funktion 23+aw
820 MX      n=aw
821 QG      IF n=1 THEN ab=0 ELSE IF n=2 THEN ab=2 ELSE ab=4
822 dX      fenster 0,30,632,197
823 1J      y=38
824 T3      TIMER OFF
825 sW      FOR i=1 TO mv
826 SA6      y=y+12
827 Ml      COLOR 3,2
828 8o      pixlocate 12,y:PRINT USING "###.";i
829 Rk      pixlocate 56,y:PRINT LEFT$(v$(i,0),31)
830 tp      kasten 320,y-8,98,11
831 f4      COLOR 3,5
832 W4      pixlocate 324,y:PRINT USING "###";st(i,ab+1);
833 M5      PRINT " von " USING "###";st(i,ab)
834 FL      kasten 430,y-8,180,11
835 qO      IF st(i,ab) THEN LINE (431,y-6)-(431+st(i,ab+1)*176
/st(i,ab),y),3,bf
836 4t4     IF y<180 THEN NEXT
837 vJ      e=0:e1=0
838 5J      FOR i=1 TO mv
839 AY6      e=e+st(i,ab)
840 nU      e1=e1+st(i,ab+1)
841 eJ4     NEXT
842 dm      IF e THEN s=e1*100/e ELSE s=0
843 AJ      kasten 12,192,280,11
844 Ds      kasten 304,192,114,11
845 EG      kasten 430,192,180,11
846 f8      kasten 12,210,134,11
847 Oj      kasten 158,210,134,11
848 eX      kasten 430,210,180,11
849 xM      COLOR 3,5
850 Vw      pixlocate 18,200:PRINT "Durchschnittlich " USING"##
# % gewußt";s
851 QH      pixlocate 308,200:PRINT USING "####";e1;
852 3I      PRINT " von " USING "####";e
853 7I      IF e THEN LINE (431,194)-(431+e1*176/e,200),3,bf
854 JB      pixlocate 22,218:PRINT "letzte Vokabel"
855 Hf      pixlocate 164,218:PRINT "nächste Vokabel"
856 nu      pixlocate 440,218:PRINT "Zurück zum Hauptmenü"
857 N5      n=1:e=1
858 iN      WHILE MOUSE(0)<0:WEND
859 Gm2     abfrage12:
860 qR4      WHILE MOUSE(0)=0:WEND
861 wR      x=MOUSE(3):y=MOUSE(4)
862 cN      IF x<12 OR x>609 OR y<210 OR y>220 THEN abfrage12
863 k3      IF x>429 THEN RETURN
864 5B      IF mv<13 THEN abfrage12
865 ZP      IF x>292 THEN abfrage12
866 Sh      IF x>151 THEN n=n+1 ELSE n=n-1
867 es      IF n=0 THEN n=1:GOTO abfrage12 ELSE IF n+11>mv THEN
n=mv-11:GOTO abfrage12
868 B1      TIMER OFF
869 2O      COLOR 3,2
870 hk      IF n<e THEN nextstat
871 gM      SCROLL (12,42)-(609,184),0,-12
872 LQ      pixlocate 12,182:PRINT USING "###.";n+11
873 KY      pixlocate 56,182:PRINT LEFT$(v$(n+11,0),31)
874 RJ      kasten 320,174,98,11
875 Nm      COLOR 3,5
876 9I      pixlocate 324,182:PRINT USING "###";st(n+11,ab+1);
877 8I      PRINT " von " USING "###";st(n+11,ab)
878 np      kasten 430,174,180,11
879 vc      IF st(n+11,ab) THEN LINE (431,176)-(431+st(n+11,ab+1)
*176/st(n+11,ab),182),3,bf
880 pv      TIMER ON
881 s1      e=n
882 ny      GOTO abfrage12
883 xv2     nextstat:
884 pp4      SCROLL (12,38)-(609,184),0,12
885 9T      pixlocate 12,50:PRINT USING "###.";n
886 2Z      pixlocate 56,50:PRINT LEFT$(v$(n,0),31)
887 9Z      kasten 320,42,98,11
888 az      COLOR 3,5
889 LX      pixlocate 324,50:PRINT USING "###";st(n,ab+1);
890 1A      PRINT " von " USING "###";st(n,ab)

```

Listing 2. (Fortsetzung)

```

891 6N      kasten 430,42,180,11
892 ZR      IF st(n,ab) THEN LINE (431,44)-(431+st(n,ab+1)*176/st
            (n,ab),50),3,bf
893 28      TIMER ON
894 5v      e=n
895 OB      GOTO abfrage12
896 UV0 REM Programm beenden
897 wq2     ende:
898 hL4     funktion 9
899 6G      loeschen
900 Lc      IF va THEN frage 0,30,56:IF aw=0 THEN RETURN ELSE anw
            speichern
901 sh      frage 0,30,57
902 M3      IF aw=0 THEN RETURN
903 972     anwspeichern:
904 Wh4     IF anw=0 THEN beenden
905 XZ      abn$(mabn,2)=LEFT$(TIME$,5)
906 3P      OPEN "BENUTZT."+LEFT$(na$(anw),3) FOR OUTPUT AS 1
907 yW      PRINT #1,mabn
908 j4      FOR i=1 TO mabn
909 s66     FOR j=0 TO 2
910 xF8     PRINT #1,abn$(i,j)
911 mr6     NEXT
912 ns4     NEXT
913 XH      CLOSE 1
914 ru      IF fae THEN GOSUB faspichern
915 EB2     beenden:
916 yM4     MENU RESET
917 j4      COLOR 3,1
918 MS      CLS
919 ZU      END
920 hw0 REM Sonderfunktionen
921 lu2     sonderfunktion:
922 pF4     funktion 10
923 oe      menuue 90,80,230,59,61
924 Zw      IF be THEN RETURN
925 X1      ON aw GOTO farbwechsel,datum,benutzung
926 qk0 REM Farben aendern
927 aQ2     farbwechsel:
928 Zj4     loeschen
929 cv      funktion 59
930 IN      fenster 0,30,348,112
931 yT      rahmen 14,44,248,19
932 Rp      FOR i=1 TO 7
933 gO6     LINE (i*34-12,48)-(i*34+16,58),1,bf
934 9E4     NEXT
935 sy      rahmen 20,47,33,13
936 Jd      FOR i=1 TO 3
937 NE6     kasten 14,i*16+56,248,11
938 ag      kasten 276,i*16+56,50,11
939 mM      FOR j=1 TO 16
940 xN8     e=248/16*j+5
941 xS      LINE (e,i*16+65)-(e+1,i*16+66),3,b
942 HM6     NEXT
943 IN4     NEXT
944 qr      kasten 150,124,176,11
945 Q0      TIMER OFF
946 Wv      COLOR 3,5
947 A1      pixlocate 284,80:PRINT "rot"
948 xz      pixlocate 284,96:PRINT "grün"
949 lh      pixlocate 284,112:PRINT "blau"
950 Sf      pixlocate 158,132:PRINT "Zurück zum Hauptmenü"
951 y4      TIMER ON
952 Zt      FOR i=1 TO 3
953 KL6     e=farb(1,i-1)*15.5
954 oO      LINE (e+4,i*16+58)-(e+7,i*16+63),4,bf
955 UZ4     NEXT
956 RR      n=1
957 vS2     abfrage13:
958 Kz4     WHILE MOUSE(0)<0:WEND
959 R2      WHILE MOUSE(0)=0:WEND
960 X2      x=MOUSE(3):y=MOUSE(4)
961 9h      IF x<14 OR x>325 OR y<44 OR y>141 THEN abfrage13
962 Xf      IF y>123 THEN IF x>148 THEN farbbeenden
963 w1      IF x>260 THEN abfrage13
964 uJ      IF y<63 THEN farbaendern
965 Qd      x=INT((x+2)/15.5)
966 bL      IF y>101 THEN y=2 ELSE IF y>85 THEN y=1 ELSE y=0
967 Th      e=farb(n,y)*15.5+4
968 we      LINE (e,y*16+74)-(e+3,y*16+79),5,bf
969 dG      farb(n,y)=x
970 It      IF n=1 THEN farb(0,y)=x

```

```

971 VA      fae=1
972 Ym      e=farb(n,y)*15.5+4
973 yf      LINE (e,y*16+74)-(e+3,y*16+79),4,bf
974 Fn      PALETTE n,f1(farb(n,0)),f1(farb(n,1)),f1(farb(n,2))
975 qc      IF n=1 THEN PALETTE 0,f1(farb(n,0)),f1(farb(n,1)),f1(
            farb(n,2))
976 Oa      GOTO abfrage13
977 j42     farbaendern:
978 4d4     x=INT((x+14)/34)
979 6b      IF x<1 THEN x=1 ELSE IF x>7 THEN x=7
980 y8      LINE (n*34-14,47)-(n*34+18,59),2,b
981 pf      LINE (n*34-13,47)-(n*34-13,59),2
982 Ou      LINE (n*34+17,47)-(n*34+17,59),2
983 yG      FOR i=0 TO 2
984 CS6     e=farb(n,i)*15.5+4
985 t5      LINE (e,i*16+74)-(e+3,i*16+79),5,bf
986 z44     NEXT
987 Wf      n=x
988 wo      rahmen n*34-14,47,33,13
989 4M      FOR i=0 TO 2
990 TE6     e=farb(n,i)*15.5
991 Z9      LINE (e+4,i*16+74)-(e+7,i*16+79),4,bf
992 5A4     NEXT
993 fr      GOTO abfrage13
994 y52     farbbeenden:
995 Pm4     FOR i=0 TO 7
996 HV6     FOR j=0 TO 2
997 308     fa(anw,i,j)=farb(i,j)
998 BG6     NEXT
999 CH4     NEXT
1000 U6     RETURN
1001 4R0 REM Abfragedaten ausgeben
1002 4Q2     datum:-
1003 mw4     loeschen
1004 dJ      funktion 60
1005 J4      IF mv=0 THEN fehler 0,30,62:RETURN
1006 GJ      IF lad=0 THEN fehler 0,30,63:RETURN
1007 qR      IF mbn=0 THEN fehler 0,30,64:RETURN
1008 wh      fenster 0,30,450,mbn*12+46
1009 S2      TIMER OFF
1010 Jf      COLOR 3,2
1011 dY      pixlocate 12,47:PRINT "Abfragedaten zur Vokabeldatei:
            "
1012 JA      kasten 260,39,174,11
1013 bO      COLOR 3,5
1014 lN      pixlocate 268,47:PRINT d$(lad)
1015 Sz      y=52
1016 P1      COLOR 3,2
1017 yY      FOR i=1 TO mbn
1018 YG6     y=y+12
1019 Yx      pixlocate 12,y:PRINT bn$(i,0)
1020 ec      pixlocate 110,y:PRINT "von " bn$(i,1) " Uhr bis
            " bn$(i,2) " Uhr"
1021 Yd4     NEXT
1022 q4      kasten 389,mbn*12+60,44,11
1023 lA      COLOR 3,5
1024 mt      pixlocate 402,mbn*12+68:PRINT "OK"
1025 Yu      COLOR 3,2
1026 BH      TIMER ON
1027 8g2     abfrage14:
1028 S74     WHILE MOUSE(0)<0:WEND
1029 ZA      WHILE MOUSE(0)=0:WEND
1030 fA      x=MOUSE(3):y=MOUSE(4)
1031 7G      IF x<389 OR x>432 THEN abfrage14
1032 9F      IF y-mbn*12<60 OR y-mbn*12>70 THEN abfrage14
1033 lD      RETURN
1034 Te0 REM Benutzungsdaten ausgeben
1035 TP2     benutzung:
1036 Jt4     loeschen
1037 CJ      funktion 61
1038 Jz      IF man=0 THEN fehler 0,30,65:RETURN
1039 rF      fenster 0,30,372,mabn*12+46
1040 xX      TIMER OFF
1041 oA      COLOR 3,2
1042 Xk      pixlocate 12,47:PRINT "Benutzungsdaten von"
1043 Km      kasten 180,39,174,11
1044 6V      COLOR 3,5
1045 84      pixlocate 186,47:PRINT na$(anw)
1046 xU      y=52
1047 uG      COLOR 3,2
1048 zK      FOR i=1 TO mabn
1049 316     y=y+12

```



```

1050 qt      pixlocate 12,y:PRINT abn$(1,0)
1051 tj      pixlocate 110,y:PRINT "von " abn$(1,1) " Uhr bis
            ";
1052 cl      IF abn$(1,2) <> "" THEN PRINT abn$(1,2) " Uhr" ELSE
            PRINT "jetzt"
1053 494     NEXT
1054 4F      kasten 310,mabn*12+60,44,11
1055 Hg      COLOR 3,5
1056 vc      pixlocate 323,mabn*12+68:PRINT "OK"
1057 4Q      COLOR 3,2
1058 hn      TIMER ON
1059 jI2     abfrage15:
1060 yd4     WHILE MOUSE(0) < 0:WEND
1061 5g      WHILE MOUSE(0)=0:WEND
1062 Bg      x=MOUSE(3):y=MOUSE(4)
1063 GO      IF x<309 OR x>352 THEN abfrage15
1064 RG      IF y-mabn*12<60 OR y-mabn*12>70 THEN abfrage15
1065 X9      RETURN
1066 ZNO     REM Unterprogramme
1067 FX      REM Menuefunktion ausgeben
1068 OZ2     SUB funktion (n) STATIC
1069 QO4     TIMER OFF
1070 Hd      COLOR 3,2
1071 3d      pixlocate 10,16:PRINT t$(n) SPACE$(40-LEN(t$(n)))
1072 v1      TIMER ON
1073 LN      END SUB
1074 rKO     REM Bildschirm loeschen
1075 up2     SUB loeschen STATIC
1076 qX4     LINE (0,23)-(632,230),1,bf
1077 PR      END SUB
1078 aFO     REM Rahmen ausgeben
1079 cF2     SUB rahmen (x,y,l,b) STATIC
1080 IW4     LINE (x,y)-(x+1-1,y+b-1),3,b
1081 TG      LINE (x+1,y)-(x+1,y+b-2),3
1082 mO      LINE (x+1-2,y)-(x+1-2,y+b-2),3
1083 VX      END SUB
1084 xSO     REM Fenster ausgeben
1085 VS2     SUB fenster (x,y,l,b) STATIC
1086 ST4     LINE (x,y+4)-(x+1-8,y+b-1),2,bf
1087 HB      LINE (x,y+4)-(x+1-8,y+b-1),3,b
1088 IV      LINE (x+1,y+5)-(x+1,y+b-2),3
1089 TI      LINE (x+1-9,y+5)-(x+1-9,y+b-2),3
1090 ay      LINE (x+8,y)-(x+1-1,y+3),3,bf
1091 wS      LINE (x+1-7,y+4)-(x+1-1,y+b-5),3,bf
1092 eg      END SUB
1093 sIO     REM Kasten ausgeben
1094 AX2     SUB kasten (x,y,l,b) STATIC
1095 Fd4     LINE (x,y)-(x+1-1,y+b-1),5,bf
1096 w4      rahmen x,y,l,b
1097 jI      END SUB
1098 UEO     REM Fenster fuer Vokabelausgabe
1099 e62     SUB vokfenster STATIC
1100 cy4     fenster 0,30,632,80
1101 wW      TIMER OFF
1102 47      pixlocate 12,46:PRINT "Fremdwort:"
1103 UH      pixlocate 12,84:PRINT "Übersetzung:"
1104 RX      TIMER ON
1105 gO      kasten 12,53,600,11
1106 jT      kasten 12,91,600,11
1107 tv      END SUB
1108 NSO     REM Ausgabemaske erstellen
1109 RM2     SUB ausgmaske STATIC
1110 Vf4     loeschen
1111 IE      fenster 0,30,350,75
1112 ks      fenster 24,116,350,75
1113 30      information 380,30
1114 O2      END SUB
1115 EwO     REM Informationsfenster ausgeben
1116 rx2     SUB information (x,y) STATIC
1117 K34     SHARED mv,mmv
1118 qL      fenster x,y,252,75
1119 Eo      TIMER OFF
1120 5R      COLOR 3,2
1121 IY      pixlocate x+12,y+18:PRINT "Information:
1122 OG      pixlocate x+12,y+34:PRINT "Vokabelanzahl gesamt:"
1123 x7      pixlocate x+12,y+48:PRINT "Vokabelnummer      :
1124 S1      kasten x+188,y+26,44,11
1125 Fk      kasten x+188,y+40,44,11
1126 Qp      COLOR 3,5
1127 8W      pixlocate x+192,y+34:PRINT USING "### #";mv
1128 Wh      pixlocate x+192,y+48:PRINT "  1"
1129 qw      TIMER ON

```

```

1130 yq      kasten x+12,y+58,220,11
1131 kO      LINE (x+13,y+60)-(x+13+mv*216/100,y+66),3,bf
1132 GH      LINE (x+14,y+60)-(x+14,y+66),4
1133 JL      END SUB
1134 fLO     REM Cursor auf Grafikkoordinaten positionieren
1135 gf2     SUB pixlocate (x,y) STATIC
1136 3E4     s=WINDOW(8)+36:t=WINDOW(8)+38
1137 5Z      POKEW s,x:POKEW t,y
1138 OQ      END SUB
1139 pEO     REM Menuefunktion
1140 DI2     SUB menue (x,y,l,at,et) STATIC
1141 ja4     SHARED aw,be
1142 2b      b=(et-at+1)*14+12:be=0
1143 3L      fenster x,y,l,b
1144 dD      TIMER OFF
1145 Uq      COLOR 3,2
1146 Y1      FOR i=at TO et
1147 cA6     pixlocate x+12,(i-at)*14+y+18:PRINT t$(i)
1148 bg4     NEXT
1149 AG      TIMER ON
1150 5P2     abfrage1:
1151 R64     WHILE MOUSE(0) < 0:WEND
1152 Y9      WHILE MOUSE(0)=0:WEND
1153 BR      xp=MOUSE(3):yp=MOUSE(4)
1154 NW      IF xp<x+1 OR xp>x+1-8 OR yp<y+4 OR yp>y+b-1 THEN
            be=1:EXIT SUB
1155 Jq      aw=INT((yp-y+4)/14)
1156 74      IF aw<1 OR aw>et-at+1 THEN abfrage1
1157 qQ      TIMER OFF
1158 K7      COLOR 4,2
1159 hH      pixlocate x+12,aw*14+y+4:PRINT t$(at+aw-1)
1160 LR      TIMER ON
1161 ln      END SUB
1162 cKO     REM Fragestellung
1163 er2     SUB frage (x,y,n) STATIC
1164 Tv4     SHARED aw
1165 M2      e=1:e1=s=1:l=0
1166 jD      WHILE e1
1167 PM6     e1=INSTR(e,t$(n),"*")
1168 uP      IF e1-e>1 THEN l=e1-e
1169 qY      IF e1 THEN e=e1+1:s=s+1
1170 J74     WEND
1171 Qw      IF LEN(t$(n))-e+1>1 THEN l=LEN(t$(n))-e+1
1172 EX      l=1*8+32
1173 11      fenster x,y,l,s*12+24
1174 9F      e=t$(n):e=y+4:e1=1
1175 81      TIMER OFF
1176 tn      WHILE e1
1177 Z16     e1=INSTR(e$,"*"):e=e+12
1178 R8      IF e1 THEN pixlocate x+12,e:PRINT LEFT$(e$,e1-1)
1179 NX      e$=MID$(e$,e1+1)
1180 TH4     WEND
1181 IF      s=s+1
1182 XR      pixlocate x+12,e:PRINT e$
1183 XE      FOR i=x+1-108 TO x+1-58 STEP 50
1184 LZ6     LINE (i,y+s*12-2)-(i+36,y+s*12+7),5,bf
1185 es      LINE (i,y+s*12-2)-(i+36,y+s*12+7),3,b
1186 g4      LINE (i-1,y+s*12-2)-(i-1,y+s*12+7),3
1187 9V      LINE (i+37,y+s*12-2)-(i+37,y+s*12+7),3
1188 FK4     NEXT
1189 PU      e=e+13
1190 Sr      COLOR 3,5
1191 7W      pixlocate x+1-98,e:PRINT "ja"
1192 RK      pixlocate x+1-55,e:PRINT "nein"
1193 sy      TIMER ON
1194 vH2     abfrage3:
1195 9o4     WHILE MOUSE(0) < 0:WEND
1196 Gr      WHILE MOUSE(0)=0:WEND
1197 J9      xp=MOUSE(3):yp=MOUSE(4)
1198 CV      IF xp<x+1-109 OR xp>x+1-22 OR yp<e-7 OR yp>e+2 TH
            EN abfrage3
1199 Bv      IF xp<x+1-63 THEN aw=1 ELSE aw=0
1200 OQ      END SUB
1201 KPO     REM Fehlermeldung
1202 KH2     SUB fehler (x,y,n) STATIC
1203 ye4     e=1:e1=s=1:l=0
1204 LF      WHILE e1
1205 ly6     e1=INSTR(e,t$(n),"*")
1206 W1      IF e1-e>1 THEN l=e1-e
1207 Sa      IF e1 THEN e=e1+1:s=s+1

```

Listing 2. (Fortsetzung)

```

1208 vj4 WEND
1209 2Y IF LEN(t$(n))-e+1>1 THEN l=LEN(t$(n))-e+1
1210 q9 l=1*8+32
1211 Pg fenster x,y,l,s*12+25
1212 lr e$=t$(n):e=y+4:e1=1
1213 kK TIMER OFF
1214 bx COLOR 3,2
1215 WQ WHILE e1
1216 Ce6 e1=INSTR(e$,"*"):e=e+12
1217 41 IF e1 THEN pixlocate x+12,e:PRINT LEFT$(e$,e1-1)
1218 OA e$=MID$(e$,e1+1)
1219 6u4 WEND
1220 es s=s+1
1221 A4 pixlocate x+12,e:PRINT e$
1222 AY LINE (x+1-54,y+s*12-2)-(x+1-18,y+s*12+8),5,bf
1223 JB LINE (x+1-54,y+s*12-2)-(x+1-18,y+s*12+8),3,b
1224 JC LINE (x+1-55,y+s*12-2)-(x+1-55,y+s*12+8),3
1225 Ap LINE (x+1-17,y+s*12-2)-(x+1-17,y+s*12+8),3
1226 2R COLOR 3,5
1227 39 e=e+14
1228 NF pixlocate x+1-44,e:PRINT "OK"
1229 SY TIMER ON
1230 Zw2 abfrage4:
1231 j04 WHILE MOUSE(0)<0:WEND
1232 qR WHILE MOUSE(0)=0:WEND
1233 tj xp=MOUSE(3):yp=MOUSE(4)
1234 WN IF xp<x+1-55 OR xp>x+1-17 OR yp<e-8 OR yp>e+2 THEN
N abfrage4
END SUB
1235 xz
1236 k10 REM Datei auswaehlen
1237 M22 SUB dateiauswahl STATIC
1238 yL4 SHARED md,be,aw
1239 g2 fenster 0,30,632,190
1240 B1 TIMER OFF
1241 20 COLOR 3,2
1242 GL pixlocate 16,47:PRINT "Vokabeldatei"
1243 lv pixlocate 250,47:PRINT "Erstellungsdatum"
1244 Js y=54
1245 Uq FOR i=1 TO md
1246 Aq6 y=y+10
1247 xV pixlocate 16,y:PRINT d$(i)
1248 PE pixlocate 250,y:PRINT da$(i)
1249 4y4 IF y<176 THEN NEXT
1250 a2 kasten 16,198,121,15
1251 JM kasten 147,198,121,15
1252 bf kasten 432,198,180,15
1253 Ts COLOR 3,5
1254 d0 pixlocate 28,208:PRINT "letzte Datei"
1255 5C pixlocate 156,208:PRINT "nächste Datei"
1256 lT pixlocate 442,208:PRINT "Zurück zum Hauptmenü"
1257 u0 TIMER ON
1258 Eh n=1:e=1:be=0
1259 Bq WHILE MOUSE(0)<0:WEND
1260 Ba2 abfrage6:
1261 Ju4 WHILE MOUSE(0)=0:WEND
1262 Pu x=MOUSE(3):y=MOUSE(4)
1263 c1 IF y<54 OR y>212 THEN abfrage6
1264 2t IF y<198 THEN aw=INT((y-47)/10):GOTO datwahl
1265 Xy IF x>431 THEN be=1:EXIT SUB
1266 y2 IF md<14 THEN abfrage6
1267 6P IF x>146 THEN n=n+1 ELSE IF x>15 THEN n=n-1
1268 CF IF n=0 THEN n=1:GOTO abfrage6 ELSE IF n+12>md THEN n
=md-12:GOTO abfrage6
TIMER OFF
1269 eE IF n<e THEN SCROLL (16,55)-(340,185),0,10:pixlocate
16,64:PRINT d$(n):pixlocate 250,64:PRINT da$(n)
1271 JK IF n>e THEN SCROLL (16,55)-(340,185),0,-10:pixlocate
16,184:PRINT d$(n+12):pixlocate 250,184:PRINT da$(n+1
2)
TIMER ON
1272 9F e=n
1273 C2 GOTO abfrage6
1274 zx
1275 Jp2 datwahl:
1276 424 IF aw+n-1>md THEN abfrage6
1277 mM TIMER OFF
1278 f1 COLOR 2,3
1279 TK pixlocate 8,54+aw*10:PRINT " " d$(aw+n-1) SPACE$(29-L
EN(d$(aw+n-1)))
pixlocate 242,54+aw*10:PRINT " " da$(aw+n-1) " "
1280 CI COLOR 3,2
1281 g2
1282 JP TIMER ON
1283 s9 aw=aw+n-1

```

```

1284 km END SUB
1285 su0 REM Vokabel auswaehlen
1286 232 SUB vokauswahl STATIC
1287 O84 SHARED aw,be,mv
1288 Tp fenster 0,30,632,190
1289 yY TIMER OFF
1290 gJ pixlocate 16,47:PRINT "Fremdwort"
1291 4B pixlocate 320,47:PRINT "Übersetzung"
1292 5e y=54
1293 sE COLOR 3,2
1294 R5 FOR i=1 TO mv
1295 xd6 y=y+10
1296 8T pixlocate 16,y:PRINT LEFT$(v$(i,0),37)
1297 Ap pixlocate 320,y:PRINT LEFT$(v$(i,1),36)
1298 r14 IF y<176 THEN NEXT
1299 hG kasten 16,198,137,15
1300 j1 kasten 163,198,137,15
1301 OS kasten 432,198,180,15
1302 Gf COLOR 3,5
1303 Am pixlocate 28,208:PRINT "letzte Vokabel"
1304 P1 pixlocate 172,208:PRINT "nächste Vokabel"
1305 8G pixlocate 442,208:PRINT "Zurück zum Hauptmenü"
1306 hn TIMER ON
1307 IU n=1:e=1:be=0
1308 yd WHILE MOUSE(0)<0:WEND
1309 6X2 abfrage8:
1310 6h4 WHILE MOUSE(0)=0:WEND
1311 Ch x=MOUSE(3):y=MOUSE(4)
1312 bc IF y<54 OR y>212 THEN abfrage8
1313 FC IF y<198 THEN aw=INT((y-47)/10):GOTO vokwahl
1314 K1 IF x>431 THEN be=1:EXIT SUB
1315 bX IF mv<14 THEN abfrage8
1316 j0 IF x>162 THEN n=n+1 ELSE IF x>15 THEN n=n-1
1317 JW IF n=0 THEN n=1:GOTO abfrage8 ELSE IF n+12>mv THEN n
=mv-12:GOTO abfrage8
TIMER OFF
1318 R1 COLOR 3,2
1319 Ie
1320 db IF n<e THEN SCROLL (16,56)-(611,185),0,10:pixlocate
16,64:PRINT LEFT$(v$(n,0),37):pixlocate 320,64:PRINT L
EFT$(v$(n,1),36)
1321 mQ IF n>e THEN SCROLL (16,56)-(611,185),0,-10:pixlocate
16,184:PRINT LEFT$(v$(n+12,0),37):pixlocate 320,184:P
RINT LEFT$(v$(n+12,1),36)
TIMER ON
1322 x3 e=n
1323 Oq GOTO abfrage8
1324 vv
1325 ng2 vokwahl:
1326 o64 IF aw+n-1>mv THEN abfrage8
1327 aA TIMER OFF
1328 Tp COLOR 2,3
1329 Gv pixlocate 8,54+aw*10:PRINT " " LEFT$(v$(aw+n-1,0),37)
SPACE$(38-LEN(LEFT$(v$(aw+n-1,0),37)))
1330 J1 pixlocate 312,54+aw*10:PRINT " " LEFT$(v$(aw+n-1,1),3
6) SPACE$(37-LEN(LEFT$(v$(aw+n-1,1),36)))
COLOR 3,2
1331 Uq
1332 7D TIMER ON
1333 gx aw=aw+n-1
1334 Ya END SUB
1335 Ga0 REM Zeichenkette eingeben
1336 yY2 SUB stringeingabe (x,y,mz,f) STATIC
1337 Bh4 SHARED z$
1338 lL TIMER OFF
1339 YO IF z$<>" " THEN COLOR 3,f:pixlocate x,y:PRINT z$;CO
LOR f,4:PRINT " " ELSE COLOR f,4:pixlocate x,y:PRINT "
"
COLOR 3,2
1340 dz
1341 GM TIMER ON
1342 Fa2 abfrage2:
1343 4U4 e$=INKEY$:IF e$="" THEN abfrage2
1344 em ac=ASC(e$)
1345 XK IF ac=8 THEN delet
1346 zG IF ac=13 THEN enter
1347 k8 IF LEN(z$)=mz THEN abfrage2
1348 Ia IF ac>128 AND ac<139 THEN e$=CHR$(ff(ac-128))
1349 wW TIMER OFF
1350 zB COLOR 3,f
1351 cJ pixlocate x+LEN(z$)*8,y:PRINT e$;
1352 Sf COLOR f,4
1353 R8 PRINT " "
1354 3F COLOR 3,f
1355 Ua TIMER ON
1356 Ry z$=z$+e$

```

```

1357 4y      GOTO abfrage2
1358 sQ2     delet:
1359 H64      IF z$="" THEN abfrage2
1360 7h       TIMER OFF
1361 bo       COLOR f,4
1362 aa       pixlocate x+LEN(z$)*8-8,y:PRINT " ";
1363 CO       COLOR 3,f
1364 cJ       PRINT " "
1365 ek       TIMER ON
1366 JQ       z$=LEFT$(z$,LEN(z$)-1)
1367 E8       GOTO abfrage2
1368 pn2      enter:
1369 Gq4      TIMER OFF
1370 JV       COLOR 3,f
1371 y7       pixlocate x+LEN(z$)*8,y:PRINT " "
1372 lr       TIMER ON
1373 BD       END SUB
1374 Vc0      REM Vokabel aufgeteilt ausgeben
1375 Jx2      SUB vokausgabe (x,y,e$) STATIC
1376 Zt4      yp=y-10
1377 Oy       TIMER OFF
1378 Fb       COLOR 3,2
1379 QK2      kommasu:
1380 2E4      e=INSTR(e$,"")
1381 g5       IF e=0 THEN yp=yp+10:IF yp<y+58 THEN pixlocate x,yp:
PRINT LEFT$(e$,40):GOTO beendenausg ELSE beendenausg
1382 2D       yp=yp+10
1383 p9       IF yp>y+58 THEN beendenausg
1384 Y5       pixlocate x,yp:PRINT LEFT$(LEFT$(e$,e-1),40)
1385 JG       e$=MID$(e$,e+1)
1386 X0       GOTO kommasu
1387 nx2      beendenausg:
1388 174      TIMER ON
1389 RT       END SUB
1390 sZ0      REM franzoesische Sonderzeichen
1391 SF2      SUB franzfont (x,y) STATIC
1392 pP4      fenster x,y,230,83
1393 eE       TIMER OFF
1394 Vr       COLOR 3,2
1395 Cb       pixlocate x+12,y+16:PRINT "franz. Zeichensatz:"
1396 t7       xp=0:yp=1
1397 nC       COLOR 3,5
1398 hF       FOR i=1 TO 10
1399 v86      xp=xp+1:IF xp=4 THEN xp=1:yp=yp+1
1400 ex       s=x+xp*70-58:t=y+yp*14+10
1401 eX       LINE (s,t)-(s+58,t+9),5,bf
1402 YQ       rahmen s,t,60,11
1403 3P       pixlocate s+6,t+8:PRINT USING "F# #: ";i;:PRINT CH
R$(ff(i))
1404 Jo4      NEXT
1405 IO       TIMER ON
1406 iK       END SUB
1407 Sy0      REM Abfrage nach Reihenfolge festlegen
1408 NX2      reihenfolge:
1409 GG4      FOR s=1 TO mv
1410 lt6       re(s)=s
1411 qv4      NEXT
1412 8k       RETURN
1413 890      REM Abfrage gemischt festlegen
1414 Up2      gemischt:
1415 JI4      FOR s=0 TO mv
1416 cf6      re(s)=0
1417 w14      NEXT
1418 zf       FOR s=1 TO INT(mv/2)
1419 yz2      zufall1:
1420 oI6      t=INT(RND(1)*mv)+1
1421 QA       IF re(t) THEN zufall1
1422 2B       re(t)=s
1423 274      NEXT
1424 F6       t=INT(mv/2)
1425 W5       FOR s=1 TO mv
1426 ho6      IF re(s)=0 THEN t=t+1:re(s)=t
1427 6B4      NEXT
1428 OO       RETURN
1429 Is0      REM Abfrage nach Koennen festlegen
1430 Ht2      koennen:
1431 WP4      IF ab=1 THEN u=0 ELSE IF ab=0 THEN u=2 ELSE u=4
1432 eW       i=0:e1=0
1433 ee       FOR s=1 TO mv
1434 Sm6      i=1+st(s,u)
1435 C9       e1=e1+st(s,u+1)
1436 Yp       vo(s)=0

```

```

1437 GL4      NEXT
1438 oP       IF i=e1=0 THEN gemischt
1439 O6        i=e1*100/1
1440 RY        e1=0
1441 nm        FOR s=1 TO mv
1442 H86        IF INT(st(s,u+1)*100/st(s,u))<=i THEN e1=e1+1:re1(
e1)=s:vo(s)=1
1443 MR4      NEXT
1444 eu2      auffuehlen:
1445 pz4      i=i+10
1446 rr        FOR s=1 TO mv
1447 mO6        IF INT(st(s,u+1)*100/st(s,u))>=i THEN zugut
1448 Bm        IF re1(e1)=s THEN zugut
1449 4a        IF vo(s)>1 THEN zugut
1450 ey        e1=e1+1:re1(e1)=s:vo(s)=vo(s)+1
1451 zO        IF e1>mv THEN aufbeenden
1452 qB2      zugut:
1453 Wb4      NEXT
1454 eT        GOTO auffuehlen
1455 Xf2      aufbeenden:
1456 114      FOR s=1 TO mv
1457 HK6      re(s)=0
1458 bg4      NEXT
1459 16        FOR s=1 TO mv/2
1460 g12      zufall2:
1461 V06      u=INT(RND(1)*mv)+1
1462 Aw       IF re(u) THEN zufall2
1463 n1        IF re(u-1)=re1(s) OR re(u+1)=re1(s) THEN zufall2
1464 Sx       re(u)=re1(s)
1465 in4      NEXT
1466 xp       u=INT(mv/2)
1467 CC        FOR s=1 TO mv
1468 AK6      IF re(s)=0 THEN u=u+1:re(s)=re1(u)
1469 mr4      NEXT
1470 4g       RETURN
1471 gO0      REM Dateinamen laden
1472 sm2      SUB datnamenlad STATIC
1473 nB4      SHARED anw,be,md
1474 PJ       be=0
1475 1W       e$="DATEIEN."+LEFT$(na$(anw),3)
1476 B1       OPEN e$ FOR APPEND AS #1
1477 sF       s=LOF(1)
1478 eO       CLOSE 1
1479 68      IF s=0 THEN KILL e$:be=1:EXIT SUB
1480 3k       OPEN "DATEIEN."+LEFT$(na$(anw),3) FOR INPUT AS 1
1481 h1       INPUT #1,md
1482 Jf       FOR i=1 TO md
1483 yO6      LINE INPUT #1,d$(i)
1484 Ft       INPUT #1,da$(i)
1485 274      NEXT
1486 mW       CLOSE 1
1487 13      END SUB
1488 TIO      REM Dateinamen speichern
1489 Ho2      SUB datnamensp STATIC
1490 v94      SHARED anw,md
1491 1X       OPEN "DATEIEN."+LEFT$(na$(anw),3) FOR OUTPUT AS #1
1492 HY       PRINT #1,md
1493 Uq       FOR i=1 TO md
1494 CT6      PRINT #1,d$(i)
1495 pQ       PRINT #1,da$(i)
1496 DI4      NEXT
1497 xh       CLOSE 1
1498 CE       END SUB
1499 n10      REM Anwendernamen laden
1500 3U2      SUB anwenderladen STATIC
1501 S14      SHARED man
1502 Jb       CHDIR "DFO:"
1503 Ro       OPEN "ANWENDER" FOR APPEND AS 1
1504 Jg       s=LOF(1)
1505 5p       CLOSE 1
1506 WJ       IF s=0 THEN anwenderinit
1507 7V       OPEN "ANWENDER" FOR INPUT AS 1
1508 oL       INPUT #1,man
1509 qP       FOR i=1 TO man
1510 zh6      LINE INPUT #1,na$(i)
1511 p8       FOR j=0 TO 7
1512 hr8      FOR k=0 TO 2
1513 GGA      INPUT #1,fa(i,j,k)
1514 Va8      NEXT

```

Listing 2. (Fortsetzung)


```

1515 Wb6      NEXT
1516 Xc4      NEXT
1517 H1       CLOSE 1
1518 dr       EXIT SUB
1519 Ss2      anwenderinit:
1520 kT4      OPEN "ANWENDER" FOR OUTPUT AS 1
1521 uD        PRINT #1,"0"
1522 M6        CLOSE 1
1523 bd        END SUB
1524 ZM0      REM Neuen Anwender einfüegen
1525 yt2      neuanwender:
1526 ep4      IF man=10 THEN man=0:RETURN
1527 hL        man=man+1:anw=man
1528 IO        na$(man)=z$
1529 zi        FOR s=0 TO 7
1530 rR6      FOR t=0 TO 2
1531 JB8      fa(man,s,t)=farb(s,t)
1532 ns6      NEXT
1533 ot4      NEXT
1534 F7        CHDIR "DFO:"
1535 zi        OPEN "ANWENDER" FOR OUTPUT AS 1
1536 f9         PRINT #1,man
1537 GB        FOR s=1 TO man
1538 IP6        PRINT #1,na$(s)
1539 Fu        FOR t=0 TO 7
1540 7d8      FOR u=0 TO 2
1541 DqA      PRINT #1,fa(s,t,u)
1542 x28      NEXT
1543 y36      NEXT
1544 z44      NEXT
1545 JT        CLOSE 1
1546 P5        OPEN "DATEIEN."+LEFT$(na$(man),3) FOR OUTPUT AS 1
1547 Kd        PRINT #1,"0"
1548 mW        CLOSE 1
1549 Rd        OPEN "BENUTZT."+LEFT$(na$(man),3) FOR OUTPUT AS 1
1550 Ng        PRINT #1,"0"
1551 pZ        CLOSE 1
1552 OO        RETURN
1553 XN0      REM Anwenderdatei speichern
1554 Tk2      faspeichern:
1555 J24      OPEN "ANWENDER" FOR OUTPUT AS 1
1556 zT        PRINT #1,man
1557 aV        FOR s=1 TO man
1558 cJ6      PRINT #1,na$(s)
1559 ZE        FOR t=0 TO 7
1560 Rx8      FOR u=0 TO 2
1561 XAA      PRINT #1,fa(s,t,u)
1562 HM8      NEXT
1563 IN6      NEXT
1564 JO4      NEXT
1565 Jn        CLOSE 1
1566 cE        RETURN
1567 Gt0      REM Benutzungsdaten laden
1568 w22      anbenutzung:
1569 Tg4      OPEN "BENUTZT."+LEFT$(na$(anw),3) FOR INPUT AS 1
1570 Gr        INPUT #1,mabn
1571 O5        FOR s=1 TO mabn
1572 X76      FOR t=0 TO 2
1573 hY8      INPUT #1,abn$(s,t)
1574 TY6      NEXT
1575 UZ4      NEXT
1576 Ey        CLOSE 1
1577 nP        RETURN
1578 9B0      REM Uhrzeit ausgeben
1579 5M2      uhr:
1580 Vr4      COLOR 3,2
1581 Qw        pixlocate 550,16:PRINT TIME$
1582 sU        RETURN
1583 qe0      REM Datas
1584 vA2      DATA Sonntag,Montag,Dienstag,Mittwoch,Donnerstag,Freitag,
           Samstag
1585 xC        DATA Vokabel - Trainer Vers. 1.0
1586 J3        DATA Diskettenbetrieb
1587 gG        DATA Vokabeln eingeben
1588 w3        DATA Vokabeln ausgeben
1589 mb        DATA Vokabeln editieren
1590 AJ        DATA Vokabeln suchen
1591 cO        DATA Vokabeln abfragen
1592 AU        DATA Statistik ausgeben
1593 CZ        DATA Programm beenden
1594 2d        DATA Sonderfunktionen
1595 wQ        DATA Vokabeldatei laden

```

```

1596 OD      DATA Vokabeldatei speichern
1597 Nb      DATA "Vokabeldatei löschen"
1598 gx      DATA Diskette wechseln
1599 4R      DATA Im Speicher suchen
1600 IZ      DATA Auf Diskette suchen
1601 kb      DATA "Vokabeln einprägen"
1602 f1      DATA "Übersetzung abfragen"
1603 8l      DATA Fremdwort abfragen
1604 JS      DATA "Schrift üben"
1605 OK      DATA Abfrage in Reihenfolge
1606 qg      DATA Abfrage gemischt
1607 JU      DATA Abfrage nach "Können"
1608 U7      DATA "Statistik von Übersetzung"
1609 45      DATA Statistik von Fremdwort
1610 A1      DATA "Statistik von Schrift-Üben"
1611 lF      DATA
1612 mG      DATA
1613 nH      DATA
1614 oI      DATA
1615 kg      DATA Auf der eingelegten Diskette*sind keine Vokabeldateien.
1616 vO      DATA Es sind keine Vokabeln zum*Sichern im Speicher.
1617 Jq      DATA "Der für Vokabeln reservierte*Speicherbereich ist voll."
1618 oi      DATA "Diese Vokabeldatei ist nicht auf der*eingelegten Diskette.*Bitte Diskette überprüfen."
1619 Ng      DATA Die Vokabeln im Speicher sind*noch nicht gespeichert worden.*Soll dennoch eine andere Datei*geladen werden?
1620 Lv      DATA Mit diesem Dateinamen existiert*schon eine andere Datei.
1621 fo      DATA Soll die Vokabeldatei wirklich*gelöscht werden?
1622 VD      DATA "Auf dieser Diskette sind keine*weiteren Eintragungen möglich."
1623 8Y      DATA Die Vokabeln im Speicher sind*noch nicht gespeichert worden.*Soll die Diskette dennoch*gewechselt werden?
1624 4w      DATA Bitte die Diskette wechseln.
1625 5s      DATA Auf der eingelegten Diskette befinden*sich keine Vokabeldateien vom angege*benen Anwender.
1626 ut      DATA Es sind keine Vokabeln zum*Ausgeben im Speicher.
1627 9S      DATA Es sind keine Vokabeln zum*Editieren im Speicher.
1628 yw      DATA Vokabeln verbessern
1629 mm      DATA "Vokabeln löschen"
1630 73      DATA "Speicher löschen"
1631 ti      DATA "Soll die Vokabel wirklich*gelöscht werden?"
1632 JQ      DATA "Die bearbeitete Vokabeldatei ist*noch nicht gespeichert worden.*Soll der Speicher dennoch*gelöscht werden?"
1633 KA      DATA "Soll wirklich der ganze*Speicher gelöscht werden?"
1634 yG      DATA Es sind keine Vokabeln zum*Suchen im Speicher.
1635 xh      DATA Ist dies die gesuchte*Vokabel?
1636 tn      DATA Die gesuchte Vokabel*ist nicht im Speicher.
1637 x8      DATA "Zurück zum Hauptmenü"
1638 nV      DATA Es sind keine Vokabeln zum*Abfragen im Speicher.
1639 4z      DATA "Es sind keine Vokabeln im Speicher,*zu denen eine Statistik ausgegeben*werden könnte."
1640 Lp      DATA Die bearbeitete Vokabeldatei ist*noch nicht gespeichert worden.*Soll das Programm dennoch*abgebrochen werden?
1641 qw      DATA Soll das Programm wirklich*beenden werden?
1642 Gk      DATA
1643 Wm      DATA "Farbwerte ändern"
1644 2i      DATA Info zu Abfragedaten
1645 aJ      DATA Benutzung des Programms
1646 tO      DATA "Es sind keine Vokabeln im Speicher,*zu denen die Abfragedaten ausgeg*eben werden könnten."
1647 x1      DATA Die Vokabeldatei ist bisher*nicht in dieser Form gespeichert worden.
1648 yq      DATA Die aktuelle Vokabeldatei*wurde noch nicht abgefragt.
1649 nw      DATA Der aktuelle Anwendername*ist unbekannt.
1650 db      DATA -1
1651 OT      DATA 232,233,234,224,226,217,219,244,231,238
1652 2v      DATA 0,0.04,0.1,0.16,0.22,0.29,0.35,0.41,0.47,0.54,0.6,0.66,0.72,0.79,0.85,0.91
1653 WO      DATA 10,9,7,10,9,7,13,13,13,1,1,1,16,1,1,12,10,3,1,10,1,1,1,15

```

(C) 1988 M&T

Listing 2. (Schluß)

Dieses Programm ist unentbehrlich beim Abtippen unserer Listings. Es hilft, Tippfehler zu vermeiden und spart viel Zeit.

Ein längeres Listing ohne Fehler abzutippen ist (fast) unmöglich. Aus diesem Grund haben wir in Ausgabe 3/88 des AMIGA-Magazins eine Eingabehilfe — den Checksummer »Checkie 42« veröffentlicht. Die hier vorgestellte Version 1.1 enthält erweiterte Funktionen und bietet mehr Komfort. Damit möglichst viele unserer Leser dieses Programm auch tatsächlich anwenden, haben wir es möglichst kurz gehalten und in einer Sprache programmiert, die alle Abtipper besitzen: Amiga-Basic. **Die Form der Listings**

Die Listingzeilen bestehen aus einer bis zu vierstelligen Zeilennummer, der zwei- beziehungsweise dreistelligen Prüfsumme und der eigentlichen Programmzeile. Beispiel:

```
10 T10 print "Hallo!"
    |
    | Prüfcode
    |
    | Zeilennummer
```

Nach einer Leerstelle im Anschluß an die Zeilennummer stehen bis zu drei Zeichen Prüfcode. Die einzelnen Zeichen können sein eine Ziffer (»0« bis »9«), ein kleiner Buchstabe (»a« bis »z«) oder ein Großbuchstabe (»A« bis »Z«). Die ersten beiden Zeichen der Prüfsumme sind der eigentliche Prüfcode. Im dritten Zeichen ist die Spaltenposition der ersten »Nicht-Leerstelle« verschlüsselt. Das ist für diejenigen Anwender interessant, welche die Struktur des Listings, also die Einrückungen durch Leerzeichen, übernehmen wollen. Ist dies nicht Ihre Absicht, können Sie die Eingabe der Checksumme schon nach den ersten beiden Zeichen mit <Return> abschließen. Bei Checkie 42 muß die Groß- und Kleinschreibung so wie im Listing abgedruckt übernommen werden.

Eingabehinweis:

Geben Sie »Checkie 42« (Version 1.1) bitte mit Checkie 42 aus der AMIGA-Ausgabe 3/88 oder 12/87 ein. Sollten Sie die alte Eingabehilfe nicht besitzen, so tippen Sie das Listing für die Version 1.1 im normalen Basic-Editor ohne Prüfsummen und Zeilennummern ab.

Der Umgang mit Checkie 42

Nach dem Start fragt das Programm nach einem Dateinamen. Unter dem angegebenen Namen speichert Checkie 42 die eingegebenen Listingzeilen ab. Existiert bereits eine Datei mit diesem Namen auf der Diskette, so haben Sie mit der Abfrage »Nur Checksummer ausgeben?« zwei Möglichkeiten:

<j> Ausgabe der Datei mit Checksumme auf den Bildschirm oder den Drucker.

<n> Einlesen der Programmzeilen aus der vorhandenen Datei und Eingabe der Checksumme mit der Tastatur.

Beide Alternativen sind gedacht für Anwender, die ein Listing nicht mit dem Zeileneditor des Checkie, sondern mit einem schnelleren und/oder komfortableren Editor ihrer Wahl — zum Beispiel dem Editor von Amiga-Basic (mit »..« »a« speichern) erfaßt haben.

Checkie 42 errechnet nach der Eingabe <j> die Prüfsummen Ihres Textes und Sie können diese dann mit dem Listing im AMIGA-Magazin vergleichen. Bei der Ausgabe auf den Bildschirm schreibt das Programm die Programmzeilen inklusive Checksummen zusätzlich in eine Datei auf Diskette mit dem Zusatz ».chk«. Diese können Sie später zum Beispiel mit dem CLI-Befehl TYPE ohne erneute Berechnung der Prüfsumme noch einmal ausgeben.

Haben Sie »Nur Checksumme ausgegeben?« mit »n« beantwortet, dann können Sie dem Programm den Vergleich überlas-

sen, in dem die Frage »Eingabe aus Datei« mit »j« beantwortet wird. Dann brauchen Sie nur noch die Checksummen eingeben. Der Checksummer holt sich die Zeile aus der angegebenen Datei statt von der Tastatur. Entspricht die eingegebene Prüfzahl nicht der errechneten, kann die Zeile gleich korrigiert werden.

Beantworten Sie obige Frage mit »n«, zählt Checkie die in der Datei vorhandenen Zeilen und wartet mit der Zeilennummer »Anzahl+1« auf die Eingabe einer neuen Zeile. Alle weiteren Eingaben hängt das Programm an die bestehende Datei an. Diese Funktion ist sinnvoll, wenn Sie ein Listing in mehreren Teilen abtippen wollen.

So tippen Sie Listings ab

Haben Sie anfangs einen Dateinamen eingegeben oder Sie hängen neue Zeilen an eine bestehende Datei an, dann arbeiten Sie im normalen Eingabemodus. Checkie 42 schlägt dabei eine Zeilennummer vor und wartet auf die Prüfsumme. Nach Eingabe derselben taucht der Cursor zwischen den zwei Trennstrichen auf. Dort muß nun die Zeile »ohne« Zeilennummer und Prüfsumme eingegeben werden. Nach Betätigen der Taste <Return> berechnet Checkie die Prüfsumme. Leerstellen vor und hinter der Programmanweisung werden ignoriert. Stimmen Programmzeile und Prüfsumme mit derjenigen im Listing überein, speichert der Checksummer die Eingabe ab und wartet auf die nächste Zeile.

Einfügemodus: Wahrscheinlich wird eine abgetippte Zeile mal einen Fehler enthalten. Checkie 42 positioniert den Cursor dann an den Anfang der Zeile und wartet auf die korrekte Eingabe. Korrekturen lassen sich mit der Backspace- oder Delete-Taste durchführen. Um Zeichenfolgen einzufügen, kann kurzfristig mit <F2> der Einfügemodus eingeschaltet werden. Dieser Modus sollte allerdings nach der Fehlerkorrektur wieder ausgeschaltet werden, da er die Eingabe verlangsamt.

Sonderfall-Prüfsumme ignorieren: Möchten Sie zum Beispiel eine Kommentarzeile

Checksummer

nicht »original« übernehmen, läßt sich trotz einer falschen Prüfsumme eine Übernahme der Zeile mit der Funktionstaste <F6> erzwingen. Sie können damit aber auch falsche Programmzeilen übernehmen. Verwenden Sie deshalb die Taste <F6> nicht gewohnheitsmäßig. Der Checksummer teilt Ihnen nach Beenden des Programms mit, wieviel Zeilen er ungeprüft übernommen hat.

Prüfsumme und Zeilennummer ändern: Natürlich kann es auch vorkommen, daß die Programmzeile zwar richtig abgetippt wurde, sich bei der Prüfsumme aber ein Fehler eingeschlichen hat. Nach Betätigen von <F1> kann die Prüfsumme korrigiert werden. Während der Eingabe der Prüfsumme läßt sich mit <F7> die vom Programm vorgeschlagene Zeilennummer verändern. Damit können Sie gezielt nur bestimmte Teile eines Listings übernehmen.

Haben Sie eine mit einem anderen Editor geschriebene Programmdatei überprüft und nur in wenigen Zeilen Fehler festgestellt, lassen sich durch Vorgabe der Nummern diese Zeilen gezielt ändern. Bei Angabe der Zeilennummer in aufsteigender Reihenfolge benötigt das Programm übrigens erheblich weniger Zeit für die Suche der Zeilen in der jeweiligen Datei. Um die versehentliche Übernahme fehlerhafter Zeilen zu verhindern, sperrt das Programm bei fehlender Übereinstimmung der Prüfsummen die Taste <F7> (Änderung der Zeilennummer).

Fehlerfrei abtippen

Eingabe beenden: Die Kombination <Ctrl-e> beendet den Programmlauf nach vollständiger Eingabe des Listings oder für eine Unterbrechung.

Am Schluß noch ein Tip für alle Leser, denen unser Basic-Editor zu langsam ist. Die Berechnung der Prüfsummen erfolgt im Unterprogramm »CalcSumme«. Dieser Teil ist sehr einfach in schnellere Sprachen, wie beispielsweise C, umsetzbar.

Wer schon einmal Fehler in einem abgetippten Listing gesucht hat, der weiß, wie frustrierend diese Arbeit sein kann. Nutzen Sie deshalb den »Checkie 42«. Sie sparen viel Zeit und müssen sich nicht dauernd auf die Suche nach tückischen Fehlern begeben. (Dieter Behlich/kn)


```

180 LB2 ELSEIF e=8 THEN
181 4h4 IF i>1 THEN
182 sz6 i=i-1 : REM <BS>
183 XY LOCATE sy,sx+1 : PRINT ". "
184 un4 END IF
185 Tn2 ELSEIF e=13 THEN
186 KZ4 IF i=AnzCsZ THEN i=AnzCsZ+1 : REM <CR>
187 cL2 ELSE
188 ra4 IF e>47 AND e<58 THEN
189 vi6 e=e-48 : REM 0-9
190 AI4 ELSEIF e>64 AND e<91 THEN
191 Rv6 e=e-55 : REM A-Z
192 jX4 ELSEIF e>96 AND e<123 THEN
193 Zx6 e=e-61 : REM a-z
194 JS4 ELSE
195 yz6 GOTO blinken : REM weder noch
196 Gz4 END IF
197 JY PRINT e$;
198 ve cs(1)=e
199 D7 i=i+1
200 A32 END IF
201 vn IF i<=AnzCsZ THEN blinken
202 P50 ESEnde:
203 K22 COLOR 1,0
204 Oh LOCATE sy,sx-15
205 oO PRINT "Checksumme:"
206 gI RETURN
207 aEO NeuZeile:
208 du2 IF FZok = wahr THEN
209 P44 NeuZeile=0
210 nn WHILE e<>13 OR NeuZeile=0
211 Ux6 LOCATE zy,1:PRINT USING "###";NeuZeile;
212 oF e=ASC(INPUT$(1))
213 gP IF e>47 AND e<58 THEN NeuZeile=NeuZeile*10+e-48
214 Jn IF NeuZeile > 9999 THEN e=8
215 In IF e=8 THEN NeuZeile=INT(NeuZeile/10)
216 vJ4 WEND
217 kh IF Checkfile THEN
218 tQ6 IF NeuZeile < Zeile THEN
219 Jy8 WHILE NOT EOF(1)
220 VGA LINE INPUT #1,e$
221 od PRINT #2,e$
222 Ip8 WEND
223 MH CLOSE 1 : CLOSE 2
224 FY GOSUB backup
225 Xa OPEN dn$+".bak" FOR INPUT AS #1
226 Tl OPEN dn$ FOR OUTPUT AS #2
227 9t Zeile=1
228 oV6 END IF
229 Yp WHILE (NeuZeile > Zeile) AND (NOT EOF(1))
230 Q8 LINE INPUT #1,e$
231 mn PRINT #2,e$
232 l7 Zeile=Zeile+1
233 C06 WEND
234 cR IF EOF(1) THEN
235 bL8 CLOSE 1
236 8l NeuZeile=Zeile
237 uN LOCATE zy,1:PRINT USING "###";NeuZeile;
238 9G Checkfile=0
239 ng6 END IF
240 oh4 END IF
241 vV Zeile=NeuZeile
242 qJ2 END IF
243 HtO RETURN
244 lH EingabeZeile:
245 n72 x=cs(AnzCsZ)
246 BW0 weiter:
247 ga2 cy=zy+INT(x/LBZeile):cx=zx+(x MOD LBZeile)
248 vy LOCATE cy,cx
249 o6 COLOR 0,1
250 mI PRINT CHR$(z(x));
251 y1 LOCATE cy,cx
252 68 IF x>apos THEN apos=x
253 xx IF Checkfile AND FZok THEN
254 wL4 IF EOF(1) THEN
255 OR6 Checkfile=0 : CLOSE 1
256 JS4 ELSE
257 ix6 e$=INPUT$(1,1)
258 Gz4 END IF
259 mV2 ELSE
260 Q24 e$=INKEY$
261 922 END IF
262 gp4 IF e$="" THEN weiter
263 OI COLOR 1,0
264 OW PRINT CHR$(z(x));
265 CF LOCATE cy,cx
266 8G e=ASC(e$)
267 UR2 IF ((e AND 127) < 32) OR e=127 THEN Controlcode
268 op IF imode THEN GOSUB insert
269 l6 PRINT e$
270 K1 z(x)=e : e=30

```

```

271 3D0 Controlcode:
272 gF2 IF e=13 OR e=10 THEN
273 lN4 RETURN
274 092 ELSEIF e=30 THEN
275 2p4 a=1
276 Ea2 ELSEIF e=29 THEN
277 034 a=LBZeile
278 6G2 ELSEIF e=31 THEN
279 DD4 a=-1
280 Gb2 ELSEIF e=28 THEN
281 Rg4 a=-LBZeile
282 9a2 ELSE
283 UX4 GOTO noCrs
284 WP2 END IF
285 hz x=x+a
286 5Z IF x>=0 AND x<LZeile THEN weiter
287 tD x=x-a
288 HB GOTO weiter
289 kEO noCrs:
290 dN2 IF e=8 THEN
291 lS4 IF x>0 THEN
292 Kk6 x=x-1
293 AM LOCATE zy+INT(x/LBZeile),zx+(x MOD LBZeile)
294 Bb FOR i=x TO apos
295 788 z(i)=z(i+1)
296 Ja PRINT CHR$(z(i));
297 Z0 IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
298 FV6 NEXT i
299 pr z(apos)=32 : PRINT " "
300 yR apos=apos-1
301 ng4 END IF
302 H12 ELSEIF e=127 THEN
303 Kk4 FOR i=x TO apos
304 GH6 z(i)=z(i+1)
305 SJ PRINT CHR$(z(i));
306 i9 IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
307 Oe4 NEXT i
308 yO z(apos)=32 : PRINT " "
309 7a apos=apos-1
310 Vy2 ELSEIF e=129 THEN
311 oI4 GOSUB EingabeSumme
312 sC x=cs(AnzCsZ)
313 9U2 ELSEIF e=130 THEN
314 513 imode=imode XOR 1
315 tU5 LOCATE 7,28
316 yZ3 IF imode THEN
317 lT5 PRINT "aus"
318 JS3 ELSE
319 YV5 PRINT "ein"
320 6a3 END IF
321 Kg2 ELSEIF e=131 THEN
322 L54 GOSUB loeschen
323 3N x=cs(AnzCsZ)
324 7Y2 ELSEIF e=134 THEN
325 bD4 RETURN
326 PR2 ELSEIF e=5 THEN
327 ZT4 FEnde=wahr
328 eG RETURN
329 F82 END IF
330 xr GOTO weiter
331 lFo insert:
332 oJ2 IF apos>x THEN
333 3o4 FOR i=apos TO x STEP -1
334 Qt6 z(i+1)=z(i)
335 q64 NEXT i
336 PH z(x)=32
337 Ny apos=apos+1
338 Kp IF apos=LZeile THEN apos=apos-1:z(LZeile)=32
339 uK FOR i=x TO apos
340 lI6 PRINT CHR$(z(i));
341 H1 IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
342 xD4 NEXT i
343 yA LOCATE zy+INT(x/LBZeile),zx+(x MOD LBZeile)
344 UN2 END IF
345 vX RETURN
346 cXO CalcSumme:
347 3s2 a=0 : b=0 : c=0
348 z2 IF e=134 THEN
349 pv4 FZok=wahr
350 WX FF6=FF6+1
351 Gz2 ELSE
352 hv4 WHILE z(apos)=32 AND apos>0
353 pI6 apos=apos-1
354 9x4 WEND
355 N1 IF apos>0 THEN
356 O36 WHILE z(c)=32

```

Listing. Der verbesserte »Checkie 42«. Bitte mit der ersten Version von »Checkie 42« oder ohne Prüfsummen und Zeilennummern eingeben.

```

357 Bt8      c=c+1
358 D16      WEND
359 Jc4      END IF
360 EJ       FOR i=c TO apos
361 Y26      j=(i-c) MOD AnzFak
362 Qs      k=(i+1-c) MOD AnzFak
363 s1      a=a+((z(i) AND 127)-32)*Faktor(j)
364 2E      b=b+((z(i) AND 127)-32)*Faktor(k)
365 Ka4     NEXT i
366 pA      cs(4)=a+Zeile-INT((a+Zeile)/62)*62
367 4N      cs(5)=b+Zeile-INT((b+Zeile)/62)*62
368 4Q      FZok=(cs(1)=cs(4)) AND (cs(2)=cs(5))
369 tm2     END IF
370 Kw      RETURN
371 pM0     Uebernahme:
372 xD2     FOR i=0 TO apos
373 ih4     PRINT#2,CHR$(z(i));
374 TJ2     NEXT i
375 VZ4     PRINT#2,""
376 5R2     Zeile=Zeile+1
377 R3      RETURN
378 N90     fertig:
379 MJ2     IF Checkfile THEN
380 ZZ4     WHILE NOT EOF(1)
381 6r6     LINE INPUT#1,e$
382 DE      PRINT#2,e$
383 cQ4     WEND
384 Ok      CLOSE 1
385 922     END IF
386 5q      CLOSE 2
387 nt      CLS
388 Rg      LOCATE 12,35
389 WL      PRINT "F E R T I G !!!"
390 I9      LOCATE 20,1
391 D1      IF FF6<>0 THEN
392 C14      PRINT "ACHTUNG!!! ";
393 9H      PRINT FF6;" Zeile(n) wurde(n) ungeprüft gespeichert."
394 IB2     END IF
395 jL      RETURN
(C) 1988 M&T

```

Listing. Der verbesserte »Checkie 42« (Schluß)

Inserenten- verzeichnis

Alcomp
17

Data Becker
2, 163, 164

Kupke Computertechnik GmbH
9

Markt & Technik Buchverlag
12/13, 20/21, 24/25, 28, 30/31

Stalter J.M.
35

Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Albert Absmeier

Leitender Redakteur: Gottfried Knechtel – verantwortlich für den redaktionellen Teil

Chef vom Dienst: Susanne Kirmaier

Redaktion: Klaus Schrödl, Ralf Sablowski

Redaktionsassistent: Andrea Kaltenhauser, Brigitte Bobenstetter, Helga Weber (202)

Mitarbeiter der Redaktion: Martin Jobst, Andreas Lietz, Christian Wolff

Alle Artikel sind mit dem Kennzeichen des Redakteurs
(kn = Gottfried Knechtel, sk = Klaus Schrödl, rs = Ralf Sablowski)
und/oder mit dem Namen des Autors/Mitarbeiters gekennzeichnet

Art-director: Friedemann Porscha

Layout: Erich Schulze (Cheflayout), Bernd Wiehl

Fotografie: Sabine Tennstaedt, Claus Troendle, Didi Zille

Titelgestaltung: Friedemann Porscha

Spritzgrafik: Ewald Standke

Computergrafik: Werner Nienstedt

Auslandsrepräsentation:

Schweiz: Markt & Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug,
Tel. 042-41 56 56, Telex: 862329 mut ch

USA: M&T Publishing Inc., 501 Galveston Drive Redwood
City, CA 94063,

Telefon: (415) 366-3600, Telex 752-351

Österreich: Markt & Technik Ges. mbH

Hermann Flaniger, Große Neugasse 28,

A 1040-Wien, Tel. 0043-222-8579455, Telex: 047-132532

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AG verlegten Publikationen und dazu, daß Markt & Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Produktionsleiter: Klaus Buck (180)

Anzeigenverkaufsleitung: »Populäre Computerzeitschriften«: Alexander Narings (780)

Anzeigenleitung: Alicia Clees (313) – verantwortlich für Anzeigen

Anzeigenformate: 1/2 Seite ist 266 Millimeter hoch und 185 Millimeter breit (2 Spalten à 86 Millimeter oder 4 Spalten à 43 Millimeter). Vollformat 297 x 210 Millimeter.

Anzeigenverwaltung und Disposition: Lisa Landthaler (233)

Anzeigen-Auslandsvertretung: England: F. A. Smyth & Associates Limited, 23a, Aylmer Parade, London, N2 0PQ. Telefon: 0044/1/3405058, Telefax: 0044/1/3419602
Taiwan: Third Wave Publishing Corp., 1-4 Fl. 977 Min Shen E. Road, Taipei 10581, Taiwan, R.O.C., Tel. 00886/2/7630052, Telefax: 00886/2/7658767, Telex: 078529335

Vertriebsleiter: Helmut Grünfeldt (189)

Leiter Vertriebs-Marketing: Benno Gaab (740)

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Straße 96, 7000 Stuttgart 1,

Bezugsmöglichkeiten: Leser-Service: Telefon (089) 46 13-366. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen.

Preis: Das Einzelheft kostet DM 16,-

Druck: SOV Graphische Betriebe, Laubanger 23, 8600 Bamberg

Urheberrecht: Alle in diesem Heft erschienenen Beiträge sind urheberrechtlich geschützt. Alle Rechte, auch Übersetzungen, vorbehalten. Reproduktionen, gleich welcher Art, ob Fotokopie, Mikrofilm oder Erfassung in Datenverarbeitungsanlagen, nur mit schriftlicher Genehmigung des Verlages. Für Schaltungen, Bauanleitungen und Programme, die als Beispiele veröffentlicht werden, können wir weder Gewähr noch irgendwelche Haftung übernehmen. Aus der Veröffentlichung kann nicht geschlossen werden, daß die beschriebenen Lösungen oder verwendeten Bezeichnungen frei von gewerblichen Schutzrechten sind. Anfragen für Sonderdrucke sind an Benno Gaab (740) zu richten.

© 1988 Markt & Technik Verlag Aktiengesellschaft

Redaktion »AMIGA«

Redaktionsdirektor: Michael M. Pauly

Vorstand: Otmar Weber (Vors.), Bernd Balzer, Werner Brodt

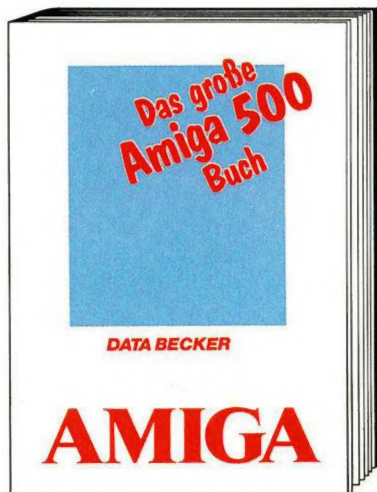
Leiter Unternehmensbereich »Populäre Computerzeitschriften«: Eduard Heilmayr, Werner Pest

Redaktionskoordination »Populäre Computerzeitschriften«: Hans-Günther Beer

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen: Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 46 13-0, Telex 5-22052

Telefon-Durchwahl im Verlag: Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089/46 13 und dann die Nummer, die in den Klammern hinter dem jeweiligen Namen angegeben ist.

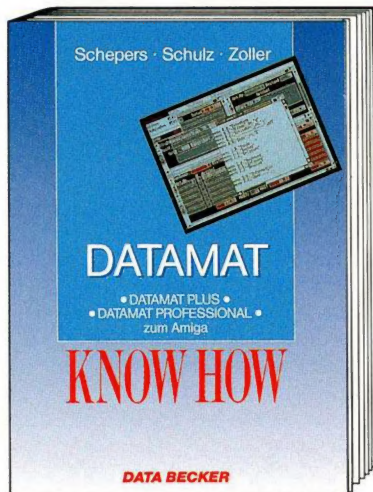
AMIGA BUCHHITS



Alles über Ihre kleine Freundin.

Wie gut das Handbuch auch sein mag, das große Amiga-500-Buch macht sich durch komplettes Detailwissen einfach unentbehrlich. Ob zur Hardware, zur Workbench oder zur Programmierung – hier finden Sie das Know-how, das einen Profi auszeichnet. Doch auch wenn Sie dieses Buch komplett durchgearbeitet haben, bleibt Ihnen das große Amiga-500-Buch als nützliches und zuverlässiges Nachschlagewerk erhalten. Ein weiterer Grund, es immer griffbereit neben dem Amiga stehen zu haben. Das große Amiga-500-Buch – die Pflichtlektüre für jeden Amiga-500-Anwender.

Das große Amiga-500-Buch
Hardcover, ca. 450 Seiten, DM 49,-
erscheint ca. 11/88



Das Know-how zu Ihrem DATAMAT-Programm.

Das Programm nach Maß: DATAMAT. In drei verschiedenen Versionen ist dieses Programm für den Amiga erhältlich. Als reine Dateiverwaltung, als einfache Datenbank und als Datenbank mit integrierter Programmiersprache. Wo liegen die Unterschiede der einzelnen Programme, was leisten sie und vor allem, wie setzt man sie optimal für eigene Anwendungen ein? Die Antworten finden Sie in „DATAMAT Know-how“. Zahlreiche Tips & Tricks runden das Ganze ab. DATAMAT Know-how – damit die Programme auch halten, was sie versprechen.

DATAMAT Know-how
ca. 400 Seiten, DM 39,-
erscheint ca. 11/88



Rund um die Datenbank Superbase.

Ob Superbase Personal II oder Superbase Professional – das große Superbase-Buch zeigt Ihnen, was diese Datenbanken im einzelnen leisten. Anfängen von der Dateidefinition, über die Dialogboxen und Schaltsymbole bis hin zur mächtigen Programmiersprache DML finden Sie hier alles, um Ihr Programm optimal für eigene Anwendungen nutzen zu können. Wie immer Sie Superbase auch einsetzen mögen, privat oder gewerblich, mit diesem Buch machen Sie mehr daraus. Das große Superbase-Buch – und Sie lernen Ihr Programm so richtig kennen.

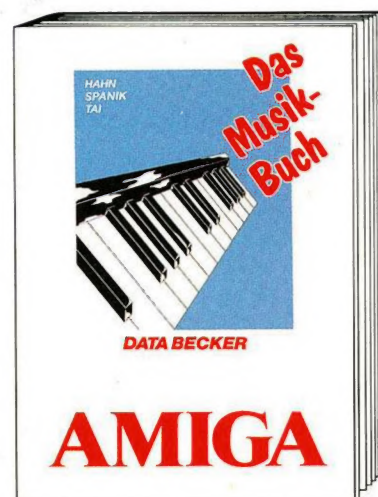
Das große Superbase-Buch
ca. 350 Seiten, DM 39,-
erscheint ca. 11/88



Perfekte Texte mit WordPerfect.

WordPerfect bietet eine unglaubliche Anzahl von Funktionen – da muß man schon bestens Bescheid wissen, um dieses Programm bis zum letzten nutzen zu können. Das große Buch zu WordPerfect kann Ihnen dabei helfen. Es ermöglicht Ihnen nicht nur einen schnellen Start, sondern vermittelt Ihnen auch jene Detailkenntnisse, die Sie brauchen, um Ihre Texte perfekt zu gestalten. Natürlich auch in diesem Buch: zahlreiche Tips & Tricks für Ihre tägliche, praktische Arbeit. Für buchstäbliche Vielschreiber einfach ein Muß.

Das große Buch zu WordPerfect.
Hardcover, ca. 320 Seiten, DM 39,-
erscheint ca. 11/88



Der Ton macht die Musik.

Zaubern Sie zarte Klänge oder heiße Rhythmen aus Ihrem Amiga. Mit dem Amiga-Musikbuch – ein Buch, das voller Musik steckt. Hier werden Sie zu einem Komponisten ausgebildet, der nicht nur die notwendigen Grundbegriffe der Musiktheorie beherrscht, sondern auch modernste Technik einzusetzen weiß. Denn in diesem Buch erfahren Sie alles zu den Musikprogrammen Sonix, Deluxe Construction Set und Audio Master. Dabei lernen Sie auch, wie Sie Sound-Sampler und MIDI-Interface professionell einsetzen. Wenn Sie mit diesem Buch gearbeitet haben, sollten Sie auch gleich der GEMA beitreten.

Amiga-Musikbuch
Hardcover, 384 Seiten, DM 49,-

COUPON

Bitte einsenden an:
DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf

HIERMIT BESTELLE ICH FÜR MEINEN AMIGA

NAME, VORNAME

STRASSE

ORT

zzgl. DM 5,- Versandkosten unabhängig von der bestellten Stückzahl
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 31 00 10

EDWORK

Das AMIGA-Werkzeug.

NEU!

Profi-Programmierer können EDwork nur bedingt empfehlen:

Erst dachten sie, EDwork sei nur irgendeiner der zahlreichen, mehr oder weniger nützlichen Editoren für den Amiga. Dann stellten sie fasziniert fest, was in dem neuen Programm steckt. Und schließlich wurde ihnen schlagartig klar, daß mit EDwork der talentierte Nachwuchs mit Macht nach oben drängen wird. Denn nach Lektüre des ausführlichen Handbuchs ist EDwork das Werkzeug für alle, die ihren ganz individuellen Editor programmieren wollen, um damit universell arbeiten zu können.

Hier sind einige der Punkte, die Profis lieber für sich behalten möchten: Mit über 120 Befehlen (und einem dazu mitgelieferten Compiler, der die Programme schneller macht) gestalten Sie sich Ihre ganz persönliche EDwork-Version – etwa mit eigenen Pulldown-Menüs. Über 65.000 Makros sind definierbar. Das Tasteninitialisierungs-Programm schreiben Sie nach Wunsch und legen damit beispielsweise kleine Programme auf einzelne Funktionstasten.

Effizient

Mitgelieferte EDwork-Hilfsprogramme für C- und Assembler-Programmierer machen Ihre Arbeit effizienter: Sie komprimieren, verschönern oder korrigieren Ihre Quelltexte. Mit "SHIFT/F10" können Sie z.B. Ihr Programm aus dem Editor heraus kompilieren lassen (etwa durch Aufruf des Aztec-C-Compilers): Eine Fehlerdatei wird geladen und der erste Fehler angezeigt. Natürlich läßt sich EDwork auch an alle anderen Programmiersprachen anpassen.

Übersichtlich

Einer der augenfälligsten Pluspunkte von EDwork wird aber die Übersichtlichkeit Ihrer Programme sein. Durch die sogenannte Faltechnik lassen Sie Module, die zur Zeit nicht Zeile für Zeile angezeigt werden sollen, einfach in den „Falten“ verschwinden. Diesen schicken „Outline“-Effekt können Sie selbstverständlich auch im Berufs- oder Privatleben nutzen. Legen Sie beispielsweise bei der Projekt- oder Terminplanung die Daten über benötigte Mitarbeiter oder Ressourcen einfach in den „Falten“ ab: Aus den Augen, aber nicht aus dem Speicher.



Preiswert

Werden Sie jedoch hellhörig, wenn Kollegen, die sich schon eine EDwork-Version gekauft haben, ihre Neuanschaffung ebenso diskret vor Ihnen verbergen wollen wie weggefaltete Programmteile. Denn EDwork ist viel zu gut und viel zu preiswert, als daß seine Vorteile nur wenigen zugute kommen sollten.

EDwork gibt es für 99,- Mark. Der Editor läuft auf allen Amiga-Rechnern der 500er-, 1000er- und 2000er-Serie.

DATA BECKER

Merowingerstr. 30 · 4000 Düsseldorf · Tel. (02 11) 31 00 10

Bitte einsenden an: DATA BECKER
Merowingerstraße 30, 4000 Düsseldorf 1

- ☐ Hiermit bestelle ich EDwork Amiga für DM 99,-
☐ per Nachnahme ☐ Verrechnungsscheck anbei

NAME, VORNAME

STRASSE

ORT